Pattern Generation for Rough Terrain Locomotion with Quadrupedal Robots: Morphed Oscillators & Sensory Feedback

THÈSE N^{O} 6518 (2015)

PRÉSENTÉE LE 6 MARS 2015

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE BIOROBOTIQUE
PROGRAMME DOCTORAL EN SYSTÈMES DE PRODUCTION ET ROBOTIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Mostafa AJALLOOEIAN

acceptée sur proposition du jury:

Prof. J. Paik, présidente du jury
Prof. A. Ijspeert, directeur de thèse
Prof. J. Buchli, rapporteur
Prof. A. Ishiguro, rapporteur
Prof. A. Kuo, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

To *Dad & Mom,*
Asiye♥,
Mohammad, Hossein, Fatemeh,
Lego, Kopol, Brownie (Ghahve'ee),
Fandogh, Felfel,

and Mohsen, Parastoo, Amir.

Acknowledgements

Completing any great milestone in life is not possible without the help, advice, and support of friends. I would like to dedicate a few paragraphs to thank people who helped me throughout my PhD both technically and personally. Though I may miss some names here, I am nevertheless grateful for all the help I have received.

First of all, I would like to thank Auke (Prof. Auke Jan Ijspeert) for the great opportunity of following my studies in Biorobotics laboratory (Biorob). Not only have you backed me up on my approach to the scientific problem of locomotion control, you have also supported me on days when I did not have results and was struggling. You took stress away from me whenever I had it and I feel indebted to you. You are the best supervisor one could ask for. Thank you!

For their constructive comments and to-the-point questions, I would like to thank the members of my thesis committee Prof. Jamie Paik, Prof. Akio Ishiguro, Prof. Jonas Buchli, and Prof. Art Kuo. It was my pleasure having you for my defense.

I have to thank the EU FP7 AMARSi project for funding most of my PhD and providing the opportunity to collaborate with knowledgeable scientists and engineers.

I thank all the past and present members of Biorob for creating such a friendly and collaborative atmosphere. Soha (Pouya), thank you for guiding me towards the right research direction and for bearing with all my annoying characteristics in the office. You are and will always be a dear friend to me. Alessandro (Crespi), thank you for all the technical help and for all the cute barcodes ;). Alex (Spröwitz), thank you for the many things you taught me about locomotion and a special thank you for making the Oncilla. Alexandre (Tuleu), a special thank you for all the support with the Oncilla simulation and hardware. Andrej (Bicanski), thank you for all the amazing coffee and sociopolitical discussions. Andrej (Gams), thank you for all the amazing experiences in Tokyo :). Cole (Simpson), thank you for all the language help and for editing this acknowledgement! Daneil (Renjewski), thank you for complying with my restaurant choice of SV ;). Florin (Dzeladini), thank you for all the help with the CPG workshop, the pizza rehearsal, and more! François (Longchamp), thank you for all the hardware support in my days of urgent need. Jérémie (Knüsel), thank you for being interested in my strange lunch discussion topics. I always liked discussing things with you :). Jesse (van den Kieboom), thank you for all the amazing and fruitful collaboration, for all the scientific help, the racklets, and for being a great friend. Kamilo (Melo), thank you for making the lab livelier :). Kostas (Karakasiliotis), thank you for the ping-pong matches (especially the ones that you lost!) and for involving me with Salamonster. Luc (Guyot), thank you for the Webots support. Luca (Collasanto), thank you for accepting Lego :). Massimo (Vespignani), thank you for all the “mi scusi”s, the sensorized

foot design, the yellowcard concert, and much more! Mehmet (Mutlu), thank you for being so kind to Lego :). Nicolas (Van der Noot), thank you for helping me with my student regarding Robotran specifics. Peter (Eckert), thank you for all your help with Oncilla and Lego. Lego loves you, and she is always happy to come to the office, which makes me happy too! :). Philippe (Müllhaupt), thank you for the in-detail discussions about the morphed oscillators. Renaud (Ronsse), thank you for welcoming me to the lab on my first days. Rico (Möckel), thank you for advising me to value my own work; it really helped a lot :). Robin (Thandiakal), thank you for being such a friendly and social figure, you are the glue of our lab. Salman (Faraji), thank you for following Soha's steps and being the advocate of model-based control in the lab. That will keep things on the right track. Sarah (Dégallier), thank you for welcoming me to the office and making it feel like a familiar and friendly place. Sébastien (Gay), thank you for all the discussion and collaboration on the Oncilla control, for the car rides, for helping me move into the new apartment, and for being a good friend. Stephane (Bonardi), thank you for all the help with thesis preparation and sorry for the snoring tractor in Tokyo! Sylvie (Fiaux), thank you for all the administrative help and friendly advice. Tomislav (Horvat), thank you for all the knowledge sharing and for making fun of the Fr. "language"! Yannick (Morel), thank you for the Fifa night experience, it was truly fun. Simon (Hauser), Tadej (Petrič), Jessica (Lanini), welcome to Biorob! Junghyun (Kim), Jérémie (Despraz), Gabriel (Chew), Stéphanie (Amati), Laetitia (Perroud), Jean-Bernard (Berteaux), Steve (Heim), Jonathan (Selz), Jan (Hermann), thank you for being amazing students. I wish you all the best for the future.

I have to thank all my friends in the Iranian community for making the life fun and preventing homesickness from kicking in. Mehran (Shahmohammadi), Hadis (Abbaspour), Zohreh (Azimi), Behzad (Dehghan), Sina (Salehian), Laleh (Makarem), Farzin (Dadashi), Arash (Arami), Mahshid (Chekini), Majid (Bastankhah), Arezou (Ghiassaleh), Mahdi (Khoramshahi), Farnaz (Eslami), thank you for all the games, invitations, dinners, and travels!

Mohammad (Khansari), Soha (Pouya), a special thank you for being great friends and helping me to feel at home when arriving in Lausanne. My life would have been different without you two and I feel lucky to have met you. Good luck with your studies and life.

I would like to thank my best three friends. Mohsen (Ramezani), thank you for all the friendly criticism and arguments. A good friend helps you understand yourself and you definitely did that. Parastoo (Alizadeh), thank you for being a nice and caring friend. You have a character that lets me easily and carefreely talk to. Amir (Firouzeh), thank you for all the PES, dinners, and for being there whenever I felt alone. I will miss you all and I deeply hope we keep the connection alive.

I would like to specially thank my parents (Baba and Maman) for teaching me to ask questions and to think about things before believing them. I am sorry that I have not always been a good student. I love you and I could not ask for anything more than what you have already given me. I hope that we will able to spend more time together in the near future.

Fatemeh, Hossein, and Mohammad, you are amazing siblings. Mohammad Hossein, Sara, Maryam, welcome to Ajallooeians! I miss all of you and I hope one day we will all live near each other. Hugs and kisses!

Lego, thank you for being with me 24/7 and for loving me unconditionally. I cannot imagine a

day without you and I hold you dear like my own child. The dog is truly the man's best friend. Kopol and Brownie (Ghahve'ee), I miss you, and it was very sad losing you. Thank you for teaching me the meaning of responsibility. Fandoq, Felfel, welcome home!
Finally and most importantly, thank you Asiye for all the love and support. Not only you are an amazing partner, but you are also my best friend. I love you infinitely and I am very happy that we can talk about everything together and I do not have to ever hide anything from you. You are the beautiful Aerith :*

Lausanne, February 2015
Mostafa Ajallooeian

Abstract

Animals are able to locomote on rough terrain without any apparent difficulty, but this does not mean that the locomotor system is simple. The locomotor system is actually a complex multi-input multi-output closed-loop control system. This thesis is dedicated to the design of controllers for rough terrain locomotion, for animal-like quadrupedal robots.

We choose the problem of *blind* rough terrain locomotion as the target of experiments. Blind rough terrain locomotion requires continuous and momentary corrections of leg movements and body posture, and provides a proper testbed to observe the interaction of different modules involved in locomotion control. As for the specific case of this thesis, we have to design rough terrain locomotion controllers that do not depend on the torque-control capability, have limited sensing, and have to be computationally light, all due to the properties of the robotics platform that we use.

We propose that a robust locomotion controller, taking into account the aforementioned constraints, is constructed from at least three modules: 1) pattern generators providing the nominal patterns of locomotion; 2) A posture controller continuously adjusting the attitude of the body and keeping the robot upright; and 3) quick reflexes to react to unwanted momentary events like stumbling or an external force impulse.

We introduce the framework of morphed oscillators to systematize the design of pattern generators realized as coupled nonlinear oscillators. Morphed oscillators are nonlinear oscillators that can encode arbitrary limit cycle shapes and simultaneously have infinitely large basins of attraction. More importantly, they provide dynamical systems that can assume the role of feedforward locomotion controllers known as Central Pattern Generators (CPGs), and accept discontinuous sensory feedback without the risk of producing discontinuous output.

On top of the CPG module, we add a kinematic model-based posture controller inspired by virtual model control (VMC), to control the body attitude. Virtual model control produces forces, and through the application of the Jacobian transpose method, generates torques which are added to the CPG torques. However, because our robots do not have a torque-control capability, we adapt the posture controller by producing task-space velocities instead of forces, thus generating joint-space velocity feedback signals. Since the CPG model used for locomotion generates joint velocities and accepts feedback without the fear of instability or discontinuity, the posture control feedback is easily integrated into the CPG dynamics. Moreover, we introduce feedback signals for adjusting the posture by shifting the trunk positions, which directly update the limit cycle shape of the morphed oscillator nodes of the CPG.

Reflexes are added, with minimal complexity, to react to momentary events. We implement

simple impulse-based feedback mechanisms inspired by animals and successful rough terrain robots to 1) flex the leg if the robot is stumbling (stumbling correction reflex); 2) extend the leg if an expected contact is missing (leg extension reflex); or 3) initiate a lateral stepping sequence in response to a lateral external perturbation.
CPG, posture controller, and reflexes are put together in a modular control architecture alongside additional modules that estimate inclination, control speed and direction, maintain timing of feedback signals, etc. We test the proposed control architecture on two simulated platforms, one that is stiff by nature and one with passive compliant legs. Using the proposed control architecture, both platforms can blindly locomote on moderately difficult rough terrains and inclined surfaces and maintain balance in the presence of external perturbations. The way pattern generation and posture control modules are implemented in this thesis is that they compete, and posture control is not done in the null-space of pattern generation. One key finding of this thesis is that such an implementation can lead to correcting an ill-parametrized pattern generator through closing the feedback loop with the posture controller. We show that this feature can be used to perform tasks like asymmetric load carriage.
Oncilla hardware robot is used as a validation tool. Utilizing the proposed control architecture with minor modifications, Oncilla can walk forwards, backwards, and turn on flat terrain, climb inclined surfaces, and prevent stumbling when trotting slowly. We exploit the leg design of Oncilla and perform uneven terrain experiments while trotting backwards, which leads to successful blind locomotion over rough terrain.
In addition to using the proposed control architecture for blind rough terrain locomotion, we use it to study the relation of feedforward and feedback modules in the locomotor system. We perform experiments to assess the effect of the proprioceptive and vestibular sensing delays on the quality of locomotion. One might expect that feedback delays would bring a constant deterioration of performance as delay increases. Counterintuitively, our experiments show that, up to a threshold, sensing delay does not greatly affect the system. Above the threshold value, sensing delays begin to visibly lower performance. These results, at a very high level of abstraction, give some insight on how the neuro-biological system might be able to handle sensing delay in neural pathways, assuming that the locomotor system is a hybrid feedforward-feedback system.

Keywords: Locomotion Control, Quadrupedal Robots, Rough Terrain, Nonlinear Oscillators, Sensory Feedback, Sensing Delay.

Résumé

Les animaux sont capables de se déplacer sur un terrain accidenté sans aucune difficulté apparente. Cela n'implique pas que le système de locomotion sous-jacent est simple: il s'agit en effet d'un système complexe en boucle fermée, présentant de multiples entrées et de multiples sorties. Cette thèse a pour but de concevoir des contrôleurs de locomotion pour terrain accidenté destinés à des robots quadrupèdes semblables à des animaux.

On a choisi en tant que but des expériences le problème de la locomotion aveugle sur terrain accidenté. Il s'agit d'un problème qui demande des corrections continues et ponctuelles des mouvements des pattes et de la posture du corps, et qui constitue ainsi un outil de test pour observer l'interaction des différents modules qui interviennent dans le contrôle de la locomotion. Pour ce qui en est du cas spécifique de cette thèse, en raison des propriétés de la plateforme robotique utilisée, il a fallu concevoir des contrôleurs de locomotion pour terrain accidenté qui ne dépendent pas de la possibilité de contrôle en couple, n'utilisent les données sensorielles que de manière limitée, et soient légers du point de vue computationnel.

On propose qu'un contrôleur de locomotion robuste, tenant compte des contraintes susmentionnées, soit composé d'au moins trois modules: 1) des réseaux locomoteurs qui fournissent les patterns nominaux de locomotion; 2) un contrôleur de posture qui corrige de façon continue l'attitude du corps et tient le robot debout; 3) des réflexes rapides pour réagir à des événements ponctuels non souhaités, comme trébucher ou une impulsion de force externe.

On introduit le cadre des morphed oscillators afin de systématiser la conception de réseaux locomoteurs sous la forme d'oscillateurs couplés. Les morphed oscillators sont des oscillateurs non-linéaires qui peuvent encoder des formes arbitraires de cycle limite, et en même temps possèdent un bassin d'attraction infiniment grand. Ce qui est plus important, est qu'ils permettent d'obtenir des systèmes dynamiques pouvant avoir le rôle de réseaux locomoteurs feedforward connus en tant que Central Pattern Generators (CPGs), et qu'ils acceptent un feedback sensoriel discontinu sans pour autant risquer de produire une sortie discontinue.

Par-dessus le module CPG, on rajoute un contrôleur de posture basé sur un modèle, inspiré par le contrôle de modèle virtuel, afin de contrôler l'attitude du corps. Le contrôle de modèle virtuel produit des forces, et à travers l'application de la méthode de transposée jacobienne, génère des couples qui sont additionnés aux couples du CPG. Cependant, nos robots n'ayant pas de possibilité de contrôle en couple, on a adapté le contrôleur de posture pour produire des vitesses dans l'espace opérationnel au lieu des forces, ainsi que des signaux de feedback des vitesses dans l'espace articulaire. Puisque le modèle de CPG utilisé pour la locomotion produit des vitesses articulaires et peut accepter un feedback sans crainte d'instabilité ou de

discontinuité, le feedback pour le contrôle de posture s'intègre aisément dans la dynamique du CPG. De plus, on introduit des signaux de feedback pour ajuster la posture en déplaçant les positions du tronc, ce qui indirectement modifie la forme du cycle limite des nœuds morphed oscillator du CPG.
Des réflexes sont ajoutés, avec une complexité minime, pour réagir à des événements ponctuels. Prenant l'inspiration des animaux et de robots se déplaçant avec succès sur des terrains accidentés, on implémente des mécanismes de feedback à impulsion simple pour fléchir la patte si le robot est en train de trébucher (réflexe de correction du trébuchage), pour étendre la patte si un contact attendu est manquant (réflexe d'extension de la patte), ou pour commencer une séquence de marche latérale en réponse à une perturbation latérale externe.
Le CPG, le contrôleur de posture et les réflexes sont assemblés dans une architecture modulaire de contrôle, en même temps que des modules additionnels pour estimer l'inclinaison, contrôler la vitesse et la direction, temporiser l'activation des signaux de feedback, etc. On teste l'architecture de contrôle proposée sur deux plateformes simulées, une de nature rigide, et une ayant des pattes élastiques passives. En utilisant l'architecture de contrôle proposée, les deux plateformes peuvent se déplacer à l'aveugle sur des terrains irréguliers modérément complexes, ainsi que sur des surfaces inclinées, et peuvent garder l'équilibre en présence de perturbations externes. Dans cette thèse, le réseau locomoteur et les modules de contrôle postural sont implémentés de manière compétitive, et le contrôle postural ne s'effectue pas dans le noyau du réseau locomoteur. Une découverte clé de cette thèse est la démonstration qu'une telle implémentation peut compenser un réseau locomoteur mal paramétré grâce à la fermeture de la boucle à l'aide du contrôleur postural. On montre que cette propriété peut être utilisée pour des tâches telles que le transport de charges asymétriques.
Le robot Oncilla est utilisé comme outil de validation. En utilisant l'architecture de contrôle proposée avec des changements peu importants, Oncilla peut marcher en avant et en arrière, ainsi que tourner sur un terrain plat, monter sur des surfaces inclinées, et éviter de trébucher lors d'un trot lent. On exploite la conception des pattes d'Oncilla en effectuant les expériences sur terrain irrégulier en marche arrière, ce qui permet d'obtenir avec succès une locomotion aveugle sur un terrain accidenté.
Ne se limitant pas à l'utilisation de l'architecture de contrôle proposée pour la locomotion aveugle sur terrain accidenté, on étudie la relation entre les modules feedforward et feedback dans le système locomoteur. On effectue des expériences pour évaluer l'effet des délais dans le retour sensoriel proprioceptif et vestibulaire sur la qualité de la locomotion. On pourrait croire que l'augmentation du délai engendre une détérioration constante des performances. Cependant, nos expériences montrent que les délais sensoriels n'affectent pas significativement le système jusqu'à une valeur limite, à partir de laquelle la performance décroit de manière évidente. Ces résultats, à un niveau d'abstraction très élevé, permettent d'avoir une certaine compréhension de comment le système neurobiologique pourrait être capable de gérer les délais sensoriels dans les chemins neuronaux, en assumant que le système locomoteur est un système hybride feedforward-feedback.

Mots clefs: contrôle de locomotion, robots quadrupèdes, terrain accidenté, oscillateurs non-linéaires, feedback sensoriel, délai sensoriel

Contents

List of Figures

List of Tables

1 Introduction

I take my dog, Lego, to our garden, let her free, and observe her running around. My apartment is on top of a hill, and our garden is quite inclined. There are bushes, a field of crops, and the ground is a mixture of asphalt and grass. But Lego does not seem to have any problem walking, trotting, galloping, turning, leaping, or balancing on any part of the garden. Of course this is not specific to Lego, and many animals are experts in dynamic locomotion over rough terrain, see Figure 1.1. Locomotion seems to be trivial.

However, from a functional standpoint, the neuro-musculo-skeletal system is far from trivial. Hundreds of muscles are working in a coordinated fashion to generate orchestrated movements. On top of that is all the circuitry generating muscle activities, which again is thousands

Figure 1.1: Different animals climbing, leaping, running and balancing on rough terrain.

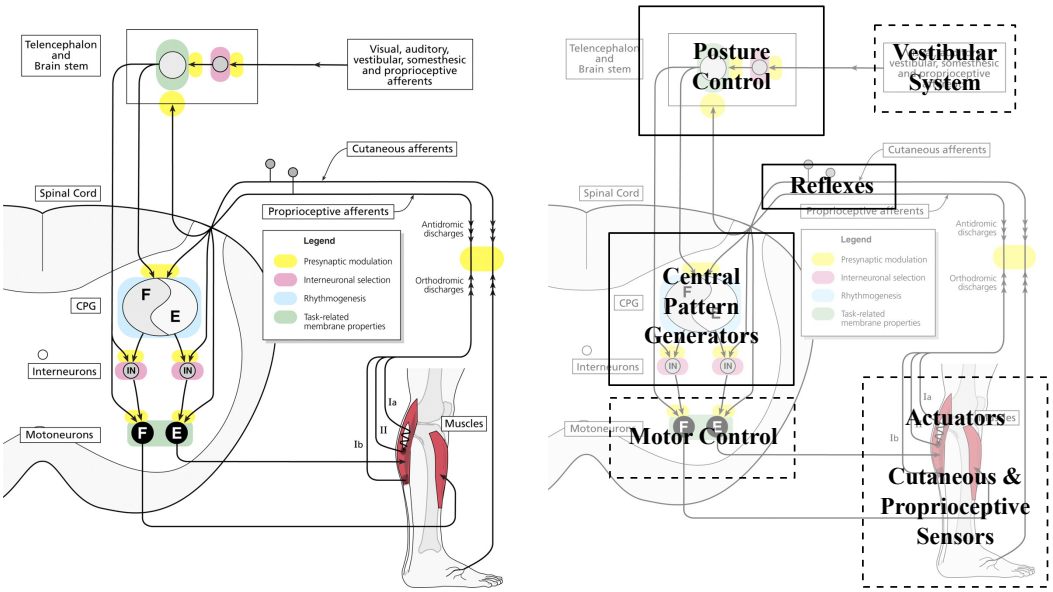


Figure 1.2: Organization of the locomotor system. *Left,* illustration taken from [Rossignol et al., 2006], which shows *some* of the sensorimotor sites interacting during locomotion. *Right,* A high-level realization of the main control modules implemented in this thesis. We design modules representing Central Pattern Generators, posture control, and reflexes. Dashed boxes are the sensorimotor modules linked to the robots used for experiments, which are given.

of neurons working in coordination. On top of the motion generation circuitry resides the high level brain control which regulates the activities to satisfy the task requirements. Not to forget that this whole control process is closed by existence of sensory feedback to all of the aforementioned levels to keep balance, regulate stiffness, etc. Figure 1.2-*left* illustrates *some* parts of this organization [Rossignol et al., 2006].

Due to the complexity of biological systems, researchers, including biologists, biomechanists, and roboticists, have studied the topic of locomotion at different levels of abstraction. Researchers have studied the organization and functionality of the locomotor system at a low-level directly concerning neurons, with the well known example of the Alan Lloyd Hodgkin and Andrew Huxley's work [Hodgkin and Huxley, 1952]. Such level of detail and low abstraction is not needed when studying locomotion at the muscle activity level. The work of Geyer et al. [Geyer and Herr, 2010] is an example which discusses how muscles should be activated and coordinated to produce locomotion. At a higher level of abstraction, locomotion can be seen as an input/output control problem, with the control outputs being position/velocity/force motor commands, neglecting the presence of the muscle system.

This thesis focuses on locomotion from a high-level abstraction standpoint, and the goal is to design controllers for locomotion of mammal-like robots. We aim at understanding which high-level control modules should be defined, and how they should interact, in order to enable robust locomotion. To do so, we will design and implement controllers for the task of blind

rough terrain locomotion. The problem of rough terrain locomotion is a suitable testbed to observe the interplay of control modules, as it involves multiple, and possibly contradicting, tasks like forward progression, balance control, maneuvering, etc. This is different from flat terrain locomotion, where a single high-level module generating repetitive patterns *can* in principle be sufficient [Spröwitz et al., 2013].

Designing robot controllers for rough terrain locomotion not only provides means for better understanding the underlying mechanisms of the locomotor system, but also is considered to be a challenging engineering problem. Such controllers should take into account the state of the robot including body rotations, accelerations, contact state, etc, and implement adaptive control strategies to deal with perturbations, uncertainties, and changes in the environment. This leads to a multi-input multi-output (MIMO) stochastic nonlinear control problem. We describe in detail in Chapter 2 successful implementations of rough terrain locomotion controllers, and their strengths, requirements, and limitations. It is worth mentioning that controller design for rough terrain locomotion is still an open question, and we provide an alternative solution to *blind* rough terrain locomotion control within the scope of this thesis.

We use simulated and hardware quadrupedal robots to test our proposed control modules and the overall control architecture. Physics-based simulation serves as the simplified environment to test scientific hypotheses by eliminating the need to build, maintain and repair a hardware robot. Moreover, simulations prevent problems like unwanted asymmetries in the body, issues arising from electronics control, etc, and make the test of the core ideas simpler. Stiff and compliant quadrupeds are modeled in simulated environments, and are used for most of the extensive tests in this thesis. Finally, since the simulations are not perfect in modeling real physics (e.g. contact modeling, linkage bendings, etc), we use hardware robots to validate the rough terrain locomotion results. We deal with several challenges when implementing our locomotion controller on the hardware robot, including imperfect sensors, imprecise calibration, limited computational power, robot's weight, actuation power/speed limitations, and more.

1.1 Questions

The main questions that we aim to answer in this thesis are:

- Which are the minimal control modules required for blind rough terrain locomotion with a quadrupedal robot?
- How to design a controller for blind rough terrain locomotion when torque-control capability is not available and the computational power is limited?
- How to systematically design pattern generators based on nonlinear oscillators to encode a desired limit cycle behavior?
- How to integrate kinematic model-based posture control into oscillator-based feedforward locomotion controllers?
- How do feedforward and feedback modules of locomotion control interact?

1.2 Approach

We want to design a modular control architecture for blind quadrupedal rough terrain locomotion. We hypothesize that three main modules should be sufficient to construct the core of a controller for rough terrain locomotion (Figure 1.2-*right*):

1. Central Pattern Generators (CPGs) generating nominal patterns of locomotion;
2. A posture controller to control the attitude and keep the body upright;
3. Reflexes to quickly react to momentary unwanted events.

We use dynamical systems theory to systematically create oscillators to take the role of abstract CPGs. We then use kinematic model-based posture control to design correction/feedback signals for CPGs. Finally, we design simple reflexes and incorporate them into the CPGs. These control modules, along with additional modules to control the interaction between them, will be put together in a modular architecture.

Throughout the design process, we pay a careful attention to the constraints imposed by the hardware robot, and keep the control modules compatible with these constraints. Constraints include computational power, available sensory information, sensitivity to sensing error, etc.

We use the proposed control architecture and assess the performance using simulated and hardware robots locomoting on different rough terrain setups including slopes, randomized terrains, etc, or in handling external perturbations. We finally use the proposed control architecture to observe the interaction between the feedforward and feedback control modules.

1.3 Contributions

In the process of creating our proposed control architecture for rough terrain locomotion, we contribute the following:

- Systematic design of nonlinear oscillators with arbitrary limit cycle shapes;
- Integration of kinematic model-based posture control and reflexes into CPGs;
- Introduction of a modular control architecture for blind rough terrain locomotion;
- Insights on the interplay of feedforward and feedback control modules in locomotion.

1.4 Organization

Part I: Preliminaries The first part provides the reader with the information needed to better understand the rest of the thesis: Chapter 2 provides a detailed overview on the state of the art. We then give a clear problem statement in Chapter 3.

Part II: Methodology The second part of this thesis contains the methodological approach taken toward the stated problem: Chapter 4 introduces the way to construct custom

oscillators which will be used to build CPGs for locomotion. Chapter 5 explains our approach toward kinematic model-based posture control and reflexes. At the end of this part, Chapter 6 describes how the control modules are put together to build the locomotion control architecture.

Part III: Experiments The final part of this thesis presents the results obtained from experiments: We illustrate the simulation results in Chapter 7. Results form the hardware experiments is then presented in Chapter 8. Finally, Chapter 9 is dedicated to a case study regarding sensory feedback delay, and its effect on locomotion control.

We summarize and discuss the results and findings of this thesis in Chapter 10. Please note that a collection of movies completing the contents of this thesis are provided online at: `http://biorob.epfl.ch/people/ajallooeian/movies`

Part I Preliminaries

2 State of the Art

Ipsa scientia potestas est (knowledge itself is power).

Francis Bacon

This thesis has been done in the framework of the AMARSi project. The locomotion control approach taken is built upon the concept of Central Pattern Generators (CPG), and we explore the task of rough terrain locomotion. Therefore we provide a summary of the AMARSi project, and detailed reviews of high-level computational CPG models and rough terrain locomotion. Moreover, we will define the concepts used throughout the thesis at their first appearances, reflecting what they mean in the context of this thesis.

2.1 AMARSi Project

In comparison to animals and humans, current robots' motor skills are rather poor and limited. There is a lack of proper adaptability, learnability, and dynamic combination of simpler motor primitives towards more complex ones. AMARSi[1] (**A**daptive **M**odular **A**rchitecture for **R**ich Motor **Ski**lls) was a large-scale EU FP7 integration project (2010-2014), with the aim to extensively improve the richness of robotic motor skills. Here richness means the ability to dynamically put a subset of simpler motor skills or adaptive modules together to realize more complex motor skills, and at the same time ensure their correct interaction, and provide procedures for inter-module and intra-module learning.

A big part of this thesis has been done under the definitions of AMARSi's Work Package (WP) 4: Adaptive Modules. The main goal of WP4 was to explore the dynamical system approach to motion generation, and design modules which can encode complex discrete

[1] Partners: Bielefeld University, EPFL, Graz University of Technology, Santa Lucia Foundation, Ghent University, Tübingen University, University of Zurich, Italian Institute of Technology, Jacobs University, and Weizmann Institute.

and/or rhythmic patterns, modulate them on-the-fly, respect a specific encapsulation for the sake of architectural compatibility, and ensure stability at all the aforementioned stages.

AMARSi project was started on April 2010, and was concluded with excellent evaluation from the EU Robotics Research Commission, on April 2014. Personally, I implemented Morphed Oscillators, as a family of dynamical systems encoding complex rhythmic patterns, and applied them to the task of rough terrain locomotion in quadrupedal robots, as a demonstration of rich motor skills. We will read more about these implementations throughout this thesis.

2.2 Central Pattern Generators for Locomotion

Central Pattern Generators (CPG) are neural circuits which are able to produce (rhythmic) patterns exogenously, i.e. in absence of sensory input [Shik et al., 1966, Grillner et al., 1998, Hooper, 2001, Ijspeert, 2008]. They have been observed in both invertebrates and vertebrates, their presence in mammals is accepted, and their existence in humans has been a point of debate [Perret and Cabelguen, 1980, MacKay-Lyons, 2002, Kiehn and Butt, 2003]. CPGs affect different rhythmic motor functions including locomotion, breathing, swallowing, etc [Dick et al., 1993, Popescu and Frost, 2002, Broch et al., 2002].

Computational models of CPGs have been implemented to model the locomotor system, and to control robots. Seminal work of [Taga et al., 1991] is one of the first studies to utilize a computational CPG model to control a (simulated) robot. This is followed by a long list of research applying computational CPG models to the robot locomotion control problem, including swimming [Crespi and Ijspeert, 2006, Zhao et al., 2006, Karakasiliotis, 2013], bipedal [Nakanishi et al., 2004, Righetti and Ijspeert, 2006b, Liu et al., 2008], quadrupedal [Fukuoka et al., 2003, Buchli and Ijspeert, 2008], and multipedal[2] [Arena et al., 2004, Inagaki et al., 2003, Sproewitz et al., 2008] locomotion. Please refer to [Ijspeert, 2008] for a detailed review.

Here we are interested to review the high-level abstract mathematical approaches to make rhythmic CPGs for locomotion control. More precisely, we will list the mathematical methodologies used in the literature to create exogenous[3] pattern generators. This excludes pattern generators which are for point-to-point reaching movements, or depend on external input including time (external clock) or sensory feedback. Table 2.1, provided later at the end of this section, summarizes the properties of different CPG models that are discussed below.

[2] Multiped: having many feet; sometimes: having more than four feet [mer, 2014a].

[3] Exogeneity helps a pattern generator to work in absence of sensory input, which can be exploited to perform early experimentation when the sensors are still not integrated into a walking robot.

Definition

Central Pattern Generator (CPG): An exogenous pattern generator without explicit dependency on time or sensory input. A CPG has internal states, and accepts sensory feedback. In general, a CPG is an autonomous (periodic) dynamical system, defined as:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{s}(\mathbf{x}) + \boldsymbol{\xi} \\ \mathbf{y} &= \mathbf{o}(\mathbf{x}) \end{aligned} \tag{2.1}$$

if it is a continuous-time system, or:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{s}(\mathbf{x}_t) + \boldsymbol{\xi} \\ \mathbf{y}_{t+1} &= \mathbf{o}(\mathbf{x}_t) \end{aligned} \tag{2.2}$$

if it is a discrete-time system. $\mathbf{x}$ is the vector of CPG's internal states, $\mathbf{y}$ is the output vector, $\mathbf{s}$ is the cascade of state transfer functions, and $\mathbf{o}$ is the cascade of the output functions. Sensory feedback can be introduced as an additive $\boldsymbol{\xi}$ term.

Before introducing the computational CPG models from the literature, we list the desired properties of a CPG model. The list includes:

- The ability to generate smooth output;
- Ease of introducing (possibly discontinuous) sensory feedback without making the output discontinuous;
- The capability of encoding arbitrary limit cycle shapes, which gives design flexibility;
- Simplicity of design/training and use.

We will discuss these items when comparing different CPG models at the end of this section.

Note: In this thesis we only address *additive* feedback introduced into the CPG dynamics. As we will see in this thesis, additive feedback is sufficient for implementation of our proposed feedback mechanisms. More complicated feedback integration (e.g. dynamics multiplied by feedback signal) is possible, but we did not find a reasonable need for extra complexity.

2.2.1 Phase Oscillators with Output Shaping

The simplest way to design an exogenous pattern generator is to have an internal clock or phase, and use a function to shape the output based on the phase value (For an unperturbed system, the variable θ for which $\dot{\theta} = \omega = \text{const.}$ is called phase [Buchli et al., 2006b]):

$$\begin{aligned} \dot{\theta} &= 2\pi\vartheta + C \\ y &= o(\theta) \end{aligned} \tag{2.3}$$

where θ is the phase state variable, ϑ is the oscillation frequency, C is the external coupling (e.g. other oscillators), and y is the output.

This idea has been implemented in different studies. For instance, [Ijspeert et al., 2007] implemented a phase driven system with trigonometric output shaping for the control of a swimming salamander robot:

$$\begin{aligned} \dot{\theta} &= 2\pi\vartheta + C \\ \ddot{r} &= a(\frac{a}{4}(R-r) - \dot{r}) \\ y &= r(1+\cos(\theta)) \end{aligned} \tag{2.4}$$

where r smoothly controls the output amplitude, R is the desired amplitude, and a is the convergence rate. There are other studies which use a similar CPG methodology, like [Spröwitz et al., 2013], where piecewise polynomials are used for output shaping.

Using phase oscillators with output shaping as CPGs has several benefits and drawbacks. First, having an explicit phase variable eases the process of phase coupling, which is needed for extension to higher dimensions (e.g. synchronized control of multiple legs / joints). Second, the freedom of choosing the output shaping allows these CPG models to generate desired output waveforms. However, the drawback of using such CPGs is the limitation of output feedback integration. If one desires to regulate the output of the CPG through feedback, then everything needs to go through the phase, θ, and possibly radius, r, dynamics to see a reflected and indirect effect on the output. This is not always straight forward to implement. Keep in mind that direct feedback on the output, $\mathbf{y} = \mathbf{o}(\mathbf{x}) + \boldsymbol{\xi}_o$, is out of question as feedback discontinuity leads to output discontinuity.

2.2.2 Low-dimensional Nonlinear Oscillators

The phase oscillators in the previous section need a function to shape the output. Here we discuss the nonlinear oscillators which do not need an output shaping function. More precisely, we discuss the case where $\mathbf{o}(.)$ is only a selector, and passes a subset of the states as the output. Hence, the shape of the output is defined by the limit cycle shape of the dynamical system, and its transient dynamics.

Let's take the example of the Hopf oscillator [Hopf, 1942, Khalil and Grizzle, 2002]. The dynamics of the Hopf oscillator is generated through the interaction of two state variables, one exciting and one inhibiting the other one:

$$\begin{aligned} \dot{x}_1 &= \gamma(\mu^2 - (x_1^2 + x_2^2))x_1 - 2\pi\vartheta x_2 \\ \dot{x}_2 &= \gamma(\mu^2 - (x_1^2 + x_2^2))x_2 + 2\pi\vartheta x_1 \end{aligned} \tag{2.5}$$

where γ is the convergence rate, and μ defines the radius of the circular limit cycle (Figure 2.1). The steady state solutions of x_1 or x_2 are sinusoidal functions, with single frequency of ϑ.

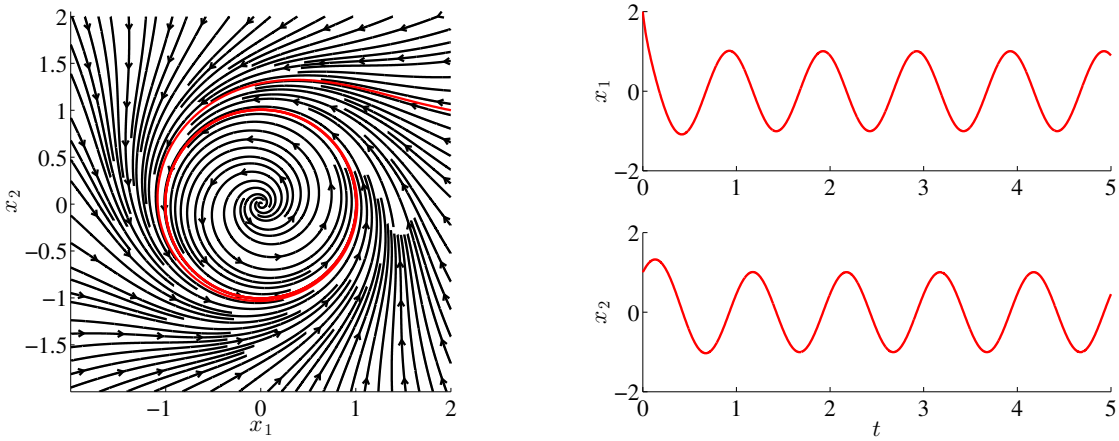


Figure 2.1: Hopf oscillator. The limit cycle is an asymptotically stable circle, and thus the steady state outputs are sine waves. x_1 and x_2 are the oscillator states, $\mu = 1$, $\gamma = 2$, and $\vartheta = 1$.

The second example is the Van der Pol oscillator [Khalil and Grizzle, 2002]. The Van der Pol oscillator is a nonlinearly damped non-conservative oscillator, defined as (with modifications):

$$\begin{aligned} \dot{x}_1 &= \vartheta(\mu x_1 - x_2 - x_1^3) \\ \dot{x}_2 &= \vartheta x_1 \end{aligned} \tag{2.6}$$

where μ controls the nonlinear damping and ϑ controls the dynamics speed (vector field magnitude, and not directly frequency). Figure 2.2 illustrates the phase portrait and time evolution of x_1 and x_2 state variables. There is one stable limit cycle, but the shape of the limit cycle cannot be written in terms of elementary functions. It is important to note that Van der Pol oscillator cannot be analytically written as a phase oscillator in polar coordinates. Very few oscillators have a phase that can be computed analytically.

Taking one further step, we slightly modify the equations of the Van der Pol oscillator with an additional state driven forcing:

$$\begin{aligned} \dot{x}_1 &= \vartheta(\mu x_1 - x_2 - x_1^3) \\ \dot{x}_2 &= \vartheta x_1 + k\ \mathrm{sign}(x_1) \end{aligned} \tag{2.7}$$

where k is the strength of the rectangular state feedback signal. Figure 2.3 illustrates the behavior of this oscillator.

Comparing the three aforementioned oscillators, we can see that the shape of the limit cycle can be defined by tailoring the state equations. Moreover, comparing Equation 2.6 with Equation 2.7, we can see that a discontinuous state feedback signal can be introduced without making the output discontinuous. However, the drawback of using such oscillators as CPGs is

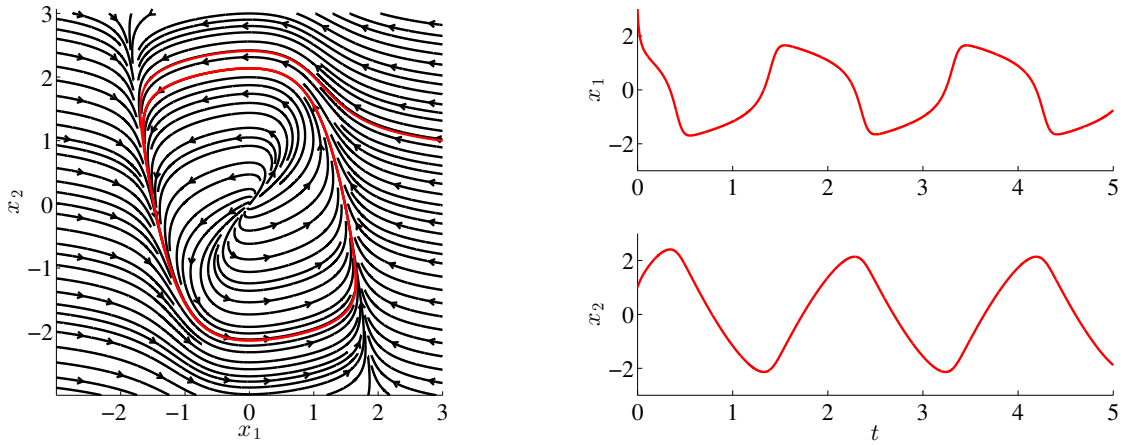


Figure 2.2: Van der Pol oscillator. The limit cycle is asymptotically stable, but the output shapes are not in form of elementary functions. x_1 and x_2 are the oscillator states, $\mu = 2$ and $\vartheta = 4$.

the design complexity: the procedure to design an oscillator with a desired limit cycle shape is not clear.

Nevertheless, this type of oscillators have been used to generate locomotion profiles whenever the profile shape is fixed or known in advance. [Righetti and Ijspeert, 2008] have demonstrated how Hopf oscillators, with additional phase resetting feedback, can be used for quadrupedal locomotion. [Righetti and Ijspeert, 2006a] shows how a set of nonlinear oscillators can be made specifically for the task of humanoid crawling. They are many more studies using fixed-shape oscillators as CPGs and incorporating sensory feedback into them, including [Buchli et al., 2006a, Buchli and Ijspeert, 2008, Matos and Santos, 2010].

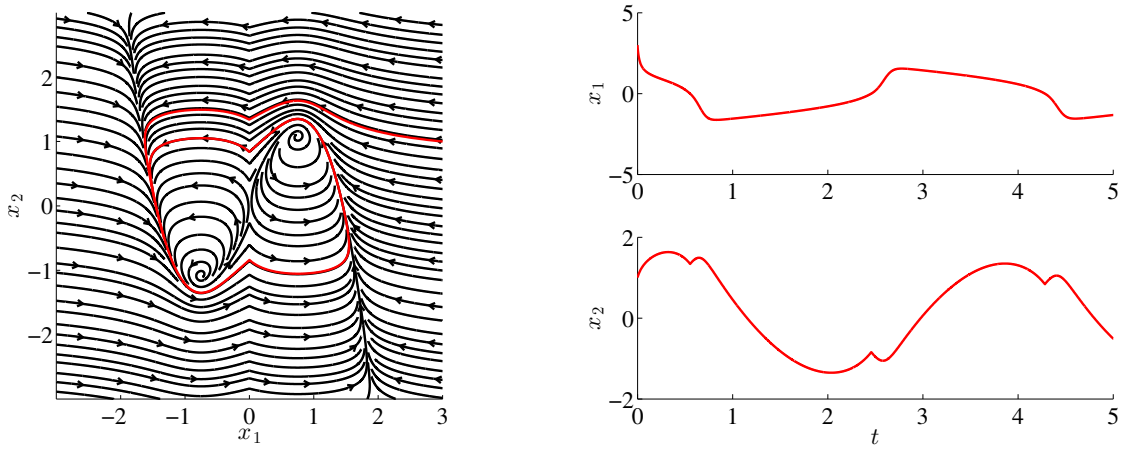


Figure 2.3: Van der Pol oscillator with state driven forcing. The shape of the limit cycle is modified by the introduction of the forcing term. Though the forcing term in discontinuous, the discontinuity is only refleted on the velocities, and the phase portrait remains continuous. Parameters are the same as the ones in Figure 2.2, and $k = -3$. Note that there are two unstable fixed points now, and further increase of k leads to disappearance of the existing limit cycle.

2.2.3 Neural Oscillators

A different way to construct computational CPG models is to use a network of neurons. Here we will not discuss the neural oscillators at a low level of abstraction, like [Hodgkin and Huxley, 1952], and we only mention the ones using a high level of abstraction.

One of the well-known models of neural oscillators is the Matsuoka oscillator [Matsuoka, 1985]. The Matsuoka oscillator is based on the observation that mutually inhibiting neurons can induce oscillations. A network of Matsuoka oscillators is implemented as:

$$\begin{aligned} \dot{x}_{1,i} &= s_i - x_{1,i} - b x_{2,i} - \sum_{j \neq i} w_{ij} x_{3,j} \\ \dot{x}_{2,i} &= \tau (x_{3,i} - x_{2,i}) \\ \dot{x}_{3,i} &= \max(0, x_{1,i} - \delta) \end{aligned} \tag{2.8}$$

where $x_{1,i}$ is the membrane potential, $x_{2,i}$ is the adaptation, $x_{3,i}$ is the firing rate, δ is the firing threshold, w_{ij} is the inhibition coupling weight between i-th and j-th neurons, s_i is the input (which can be set to a constant to respect exogeneity, or can be taken from other neurons outputs), and b and τ are time constants.

The behavior of a Matsuoka oscillator depends on the choice of the parameters and the coupling scheme used. [Degallier and Ijspeert, 2010] shows an example where a network of three bilaterally coupled neurons generates sustained oscillations. They show that if the couplings are replaced by unilateral couplings, with the same parameters, the network oscillates in a completely different regime. [Matsuoka, 2011] gives a partial solution to predict the behavior of the Matsuoka oscillators knowing the parameter values, but the complete answer, including the frequency response, is not known.

Many robot controllers have utilized the Matsuoka oscillators [Taga et al., 1991, Liu et al., 2008, Endo et al., 2008, Wu and Ma, 2010]. One interesting example is [Fukuoka et al., 2003, Kimura et al., 2007b] where they draw rules on how to incorporate sensory feedback into the Matsuoka oscillators for successful quadrupedal walking over rough terrain. This example will be discussed more in detail in Section 2.3.

Another general way to create neural oscillators (typically high-dimensional) is to utilize recurrent neural networks (RNN), and several studies have employed this idea [Ruiz et al., 1997, Pearlmutter, 1989, Doya and Yoshizawa, 1989, Townley et al., 2000, Kuroe and Miura, 2005, Leclercq et al., 2005, Kuroe and Lima, 2006, Jouffroy, 2008]. A recurrent neural network can be represented as:

$$\begin{aligned} \mathbf{x}_{t+1} &= (1-\lambda)\mathbf{x}_t + \lambda\, \sigma_x(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_f \mathbf{y}_t + \mathbf{W}_b) \\ \mathbf{y}_{t+1} &= \mathbf{W}_o\, \sigma_y(\mathbf{x}_{t+1}) \end{aligned} \tag{2.9}$$

where $\mathbf{x}_t$ is the state vector, λ is the leak rate, σ_x is the state transfer function (normally a sigmoid function), and σ_y is the output transfer function. $\mathbf{W}_x$, $\mathbf{W}_f$, $\mathbf{W}_b$ and $\mathbf{W}_o$ are respectively the weight matrices for the state, feedback, bias and the output.

The main difficulty in using recurrent neural networks is the process to tune the weight matrices described above, to obtain a desired output. Different learning methods exist to train RNNs, including gradient based methods [Pearlmutter, 1989, Pineda, 1987, Pearlmutter, 1995] and global optimization methods [Juang, 2004], but they normally have to deal with either premature local minima, or long training time. It is worth mentioning that RNNs can exhibit chaotic behavior, and have scaling issues, which make them difficult to train with big numbers of neurons. Also it is not trivial to assess the asymptotic stability of a resulting network, and predict its behavior [Ruiz et al., 1997].

Echo state networks [Jaeger, 2001, 2002, Jaeger and Haas, 2004] and reservoir computing approaches [Schrauwen et al., 2007, LukošEvičIus and Jaeger, 2009] nicely address the aforementioned difficulties. The idea is to *exploit* the weight matrices instead of tuning them. What echo state and reservoir computing approaches do is: 1) create a big RNN; 2) randomize the weight matrices while satisfying predefined numeric conditions (namely, keep the biggest eigenvalue of the state weight matrix smaller than one, and satisfy some desired connection sparsity); 3) use a linear transfer function for readout and train it using linear regression. The essence of such approaches is to create a pool of various dynamics which can include *almost* everything, and then simply choose a linear combination of them. Reservoir RNNs are capable of creating nonlinear oscillators able to encode limit cycles of arbitrary shapes, and have also been applied to the locomotion control problem [wyffels and Schrauwen, 2009].

Neural oscillators are proper choices for CPGs if one intends to hypothesize about the internal activity of neural circuits generating rhythmic activities. They also benefit from a wide range of dynamics that they can encode, and they are suitable tools to study emergent dynamics. However, the limitation of using neural oscillators as CPGs is threefold: 1) Temporal adjustments of a neural oscillator is not straight forward as there is no state variable representing phase/time. For the case of RNNs, this has been recently addressed in [wyffels et al., 2014]; 2) Integration of sensory feedback should go through the state dynamics, and it is not easy to predict the effect of state feedback in a large system. Moreover, there is no guarantee that state feedback keeps such RNNs stable; 3) A RNN seems to be an overkill if the sole purpose is to create a CPG with arbitrary limit cycle, because the computational time to evaluate them is high, due to the high number of states.

2.2.4 Arbitrary Limit Cycle Oscillators

The last set of computational CPG models that we discuss here are the models which are able to encode arbitrary limit cycle shapes. These are additional to the approaches discussed in the previous section.

Nonlinear oscillators with an arbitrary limit cycle shape can be constructed by using fitting tools on data-driven generated vector fields. Okada et al. [Okada et al., 2002] proposed a method to represent the desired trajectory as a limit cycle and define a corresponding vector field in the vicinity of each data point directed towards the limit cycle. They use a polynomial approximation of the vector field to create the dynamical system encoding the desired limit cycle. Similarly, [Ajallooeian et al., 2012] use the desired trajectory as the limit cycle and define margins of attraction based on the Poincaré-Bendixson theorem [Guckenheimer and Holmes, 1983]. They map the desired periodic trajectory on the limit cycle of a stable oscillator (and vice versa) to strengthen the stability properties, and employ feedforward neural networks to create the maps. Both [Okada et al., 2002] and [Ajallooeian et al., 2012] create dynamical systems which are trained to be (asymptotically) stable in the margins defined around the limit cycle, but not necessarily outside of these margins.

Another method to create a nonlinear oscillator with a desired limit cycle shape is to use a pool of oscillators and combine their outputs. Righetti et al. [Righetti and Ijspeert, 2006b] presented Adaptive Frequency Oscillators and used a pool of them to create programmable CPGs. They utilize a dynamic estimation of the phases of Fourier harmonics constructed from a pool of adaptive Hopf oscillators [Righetti et al., 2006, Buchli et al., 2006a], and build the desired dynamics. The proposed method in [Righetti and Ijspeert, 2006b] benefits from the fact that no external training procedure is needed, and the model encodes the input signal in an online self-learning manner. Nevertheless, the difficulty of using a pool of oscillators is two fold: 1) the output of the weighted sum of local oscillators, which makes the integration of the feedback difficult (also known as the credit assignment problem [Minsky, 1961]); 2) the final output shape of such system is limited, unless an infinite number of oscillators is used.

The last method that we discuss here is the Dynamical Movement Primitives (DMP) [Ijspeert et al., 2003, 2002a,b, 2013]. The main idea of the DMP is to use a virtual linear spring and force it such that the desired output is generated. If the forcing term is periodic, then the obtained DMP is a nonlinear oscillator with an arbitrary limit cycle shape. Rhythmic Dynamical Movement Primitives are formulated as:

$$\begin{aligned} \dot{\theta} &= \tau^{-1} \\ \dot{x}_1 &= \tau^{-1}\alpha(\beta(g - x_2) - x_1) + f(\theta) \\ \dot{x}_2 &= \tau^{-1} x_1 \end{aligned} \tag{2.10}$$

where α, β, and τ are time constants, g is the oscillation midpoint, ϑ is the oscillation frequency, and $f(\theta)$ is the phase dependent forcing term. Rhythmic DMP gives a nice formulation of a phase oscillator with an arbitrary limit cycle shape, and in essence it is a forced second order periodic dynamical system. Rhythmic DMP has most of the desired features like simplicity, and the capability of smooth feedback integration. We will see in Chapter 4 that the rhythmic DMP is a specific realization of our proposed morphed oscillators.

2.2.5 Summary

We discussed different mathematical approaches to constructing computational CPG models. Some of these approaches are originating from biological inspiration, while the others are purely abstract mathematical systems. Table 2.1 compares these different approaches. Though there are many different ways of constructing oscillators acting as CPGs, we can see that most of the introduced methods lack some of the favored properties, mentioned in the beginning of this section. Abstract mathematical models have few state variables and are easy to implement, but lack design flexibility. Neuronal models nicely capture neuron activities, but are rather complex to design. RNN models introduce a vast horizon of possibilities, but are not easy to construct, train and analyze. Data driven oscillators create the possibility to have arbitrary limit cycle shapes, but cannot guarantee a large basin of attraction. Pool of oscillators propose an easy way to construct arbitrary limit cycle shape oscillators, but are not suitable for direct feedback integration. Among all, rhythmic DMPs seem to possess most of the desired properties. We will propose the morphed oscillators (also included in the table) later in Chapter 4, which introduces a family of nonlinear oscillators with all the desired properties mentioned in Table 2.1, and includes the rhythmic DMP as one possible realization.

2.3 Quadrupedal Rough Terrain Locomotion

Here in this section we will discuss different control approaches towards the problem of rough terrain locomotion. We focus on mammal-like quadrupedal robots, as this thesis will only address this subset of legged machines.

We first explain the concept of rough terrain locomotion and categorize it in different subsets. We then mention the platforms which have been used to perform rough terrain locomotion. After that we analyze different high-level (task/behavior level) modules used in different control methodologies, and cover controller specific details.

2.3.1 Cases of Rough Terrain Locomotion

Rough terrain locomotion simply is locomotion over uneven terrestrial substrate. This includes inclined surfaces, randomized height maps, stairs, rocky terrain, etc. Based on the difficulty and terrain geometry, rough terrain locomotion can be divided into two categories: 1) visually-guided; and 2) blind.

Definition

Visually-guided rough terrain locomotion: Rough terrain locomotion that *needs* exteroception (e.g. vision or a given terrain map) for successful performance. For this case, exteroception is needed to plan the control inputs prior to walking on a specific part of the terrain. An example is walking on a terrain which contains holes wider than the stride length.

Name	Smooth output	Feedback on output	ALC	Basin of attraction	Simplicity
Phase-oscillator with output shaping	no	discontinuity	yes	-	easy to construct and use
Hopf oscillator / Van der Pol oscillator	yes	direct	no	infinitely large	easy to construct and use
Matsuoka oscillator	yes	direct	yes	limited	difficult to construct ALC
Reservoir RNN	yes	indirect	yes	limited	moderately easy to construct
Data-driven vector fields	yes	direct	yes	limited	moderately difficult to construct
Pool of oscillators	yes	indirect	yes	infinitely large	easy to construct and use
Rhythmic DMPs	yes	direct	yes	infinitely large	easy to construct and use
Morphed oscillators (Chapter 4)	yes	direct	yes	infinitely large	easy to construct and use

Table 2.1: Comparison between different methods to construct computational CPG models. ALC: Arbitrary Limit Cycle.

Definition

Blind rough terrain locomotion: Rough terrain locomotion relying *only* on proprioceptive sensors. This means that the control input should be adjusted reactively and on-the-fly as the robot goes over the rough terrain, and no pre-planning is possible. An example is locomotion over natural substrate like uneven soil or grass.

Figure 2.4 clarifies the aforementioned categorization. In the left column we see examples of the visually-guided case. These examples usually contain parts where probable failure leads to considerable damage. Here successful locomotion needs visual servoing [Bonin-Font et al., 2008] of the environment, in-advance planning of the footholds, and accurate execution of them. The right column in Figure 2.4 depicts examples of the blind case. Specifically, this includes examples of blind dogs running on uneven terrain [jak]. Blind rough terrain locomotion can be accomplished by exploiting continuous *online* posture adjustments, quick reflexes to adjust the stepping behavior to prevent unwanted situations like stumbling, and recovery actions like fall recovery.

It is worthwhile to mention that the categorization of whether locomotion on a given terrain can be accomplished blindly or not greatly depends on the size of the animal/robot and the kinematic and dynamic capabilities of it. For example, locomotion on sticks (Figure 2.4, bottom-left) can be done blindly if the size of the foot is bigger than the spacing between the sticks. As another example, successful locomotion facing obstacles (Figure 2.4, bottom-right) needs exteroception if the obstacle height is much bigger than the typical foot clearance.

2.3.2 Platforms

In this section we introduce the quadrupedal platforms which have been used for rough terrain locomotion (Figure 2.5). We only mention mammalian-like quadrupedal robots to respect the focus of this thesis, and skip sprawled (salamander-like and insect-like) quadrupeds. There are other well known quadrupeds, including MIT-Cheetah [Seok et al., 2013], that have not yet been tested for rough terrain locomotion[4], and we will not mention them in the list below. Moreover, the list here does not include the Oncilla robot, as it will be discussed in detail in Chapter 8. The list is sorted in an alphabetical order. The following list is descriptive, and analysis of pros and cons of the platforms and the control methodologies is provided at the end of this chapter.

Note: Cheetah-cub robot is included in the list of rough terrain locomotion robot regardless of its very limited capability on rough terrain. We included Cheetah-cub only because the Oncilla robot is based on it, and Oncilla is used for hardware experiments in this thesis.

[4] At the time of writting, there are online movies demonstrating MIT-Cheetah's outdoor locomotion skills, but no reference publications for these demonstrations.

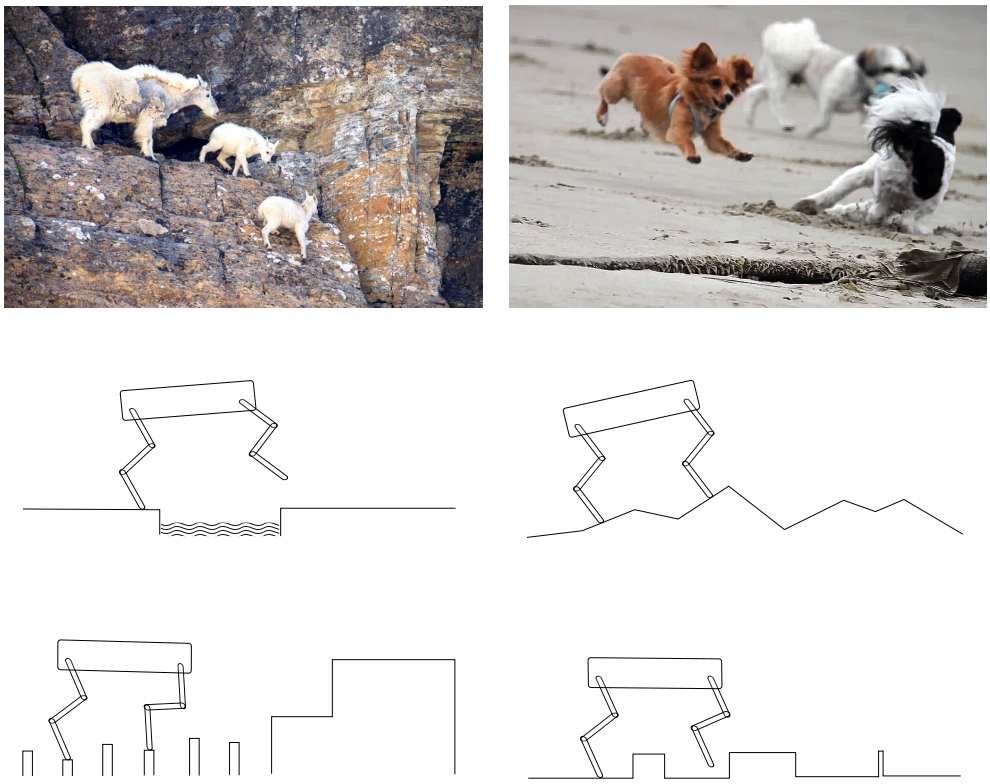


Figure 2.4: Visually-guided vs. blind rough terrain locomotion. Top-left) mountain goats climbing rocks. Top-right) a *blind* Chihuahua-Papillion mix running on the beach. Bottom-left) Examples of visually-guided rough terrain locomotion. Observation is needed to do accurate and planned foot placement, or take leaps bigger than the usual stride length. Bottom-right) Examples of blind rough terrain locomotion. The robot can locomote on these terrains, and adjusts its posture continuously. Quick reflexes are needed to prevent stumbling.

BigDog

BigDog [Playter et al., 2006, Raibert et al., 2008], designed and manufactured by Boston Dynamics, is the successor of the MIT LegLab quadrupeds, designed specifically for outdoor rough terrain locomotion. BigDog weights about 90[Kg] and the dimensions are $1 \times 1 \times 0.3$[m] (height/length/width) [Playter et al., 2006]. BigDog uses a water-cooled two stroke combustion engine capable of 17[hp] power delivery. Each leg has 4 degrees of freedom, one passive linear pneumatic compliance in the lower leg, and three active hydraulically actuated joints for hip abduction/adduction (AA), hip protraction/retraction (PR), and knee flexion/extension (FE). BigDog is equipped with about fifty different sensors. Proprioception includes joint angle and force sensing, IMU, and contact force sensing. Exteroception includes stereo vision, ambient temperature sensor, proximity sensors and LIDAR for human leader following. BigDog is also equipped with sensors to monitor homeostasis, e.g. hydraulic pressure, engine temperature, etc. A PC104 running real time QNX is used for control, with a low level control rate of 0.5-1[KHz].

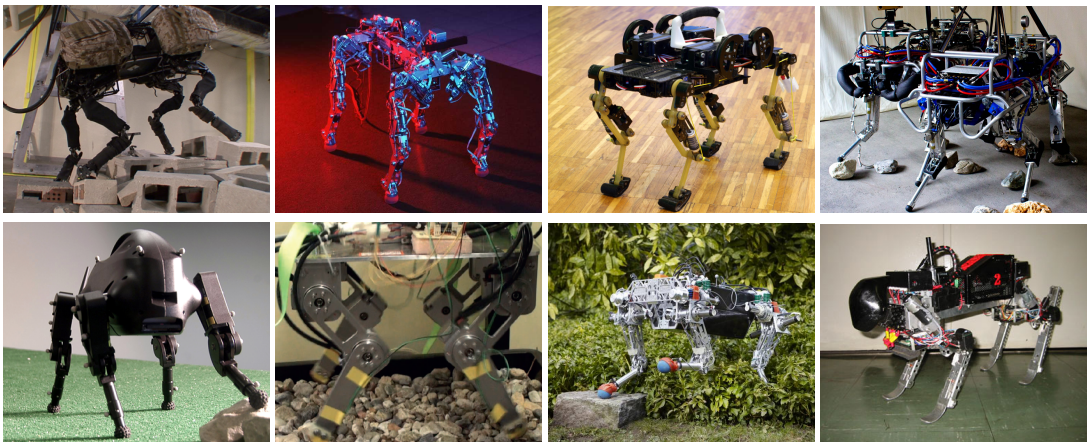


Figure 2.5: Quadrupedal robots for rough terrain locomotion. From top-left: BigDog, BISAM, Cheetah-cub, HyQ, LittleDog, RoboCat-1, StarlETH, Tekken2.

BISAM

BISAM (Biologically InSpired wAlking Machine) is developed by Forchungszentrum Informatik Karlsruhe [Berns et al., 1999]. BISAM is about 23[Kg] and the standing height is 0.7[m]. The main body is multi-segment and has five active degrees of freedom. Each leg has four degrees of freedom, namely hip AA, hip PR, knee FE and ankle FE. All joints except the ankle joints are driven by Faulhaber 12[V] 3042 DC servo motors. The ankle joints are actuated by smaller and faster but less powerful Faulhaber 2342 DC servo motors. BISAM is equipped with joint encoders, 3D force sensors, two inclinometers and three gyroscopes. A distributed computer architecture is used for control. C-167 microcontrollers along custom made sensor boards perform the low-level processing. High-level control is done on a PC104, connected via CAN bus to the low-level controller. Linux with a real time kernel is used as the operating system.

Cheetah-cub

Cheetah-cub is a light weight quadrupedal robot made by the Biorobotics laboratory, EPFL [Spröwitz et al., 2013]. Cheetah-cub weights 1.1[Kg] and the dimensions are $0.16 \times 0.2 \times 0.1$[m] (height/length/width). The main body is a rigid carbon fiber plate. Legs follow a three segmented spring loaded pantograph (SLP) four bar mechanism which keeps the first and third leg segments parallel. As introduced in [Spröwitz et al., 2013], legs are upgraded to an advanced version (ASLP) where one segment of the four bar mechanism is replaced with a linear spring. A compliant foot is also added in the ASLP design. The segment lengths of front and hind legs are different, but the overall leg lengths are about the same. In the ASLP design, each leg has two passive degrees of freedom and two active ones. Hip PR is actuated directly on the axis, while leg FE is done through a cable-clutch transmission, and the motor is located proximally to reduce the leg inertia. All the active degrees of freedom are actuated by Kondo KRS2350 ICS RC servo motors, with a tethered power input of 8-14[V]. All the computation is

done onboard on a RoBoard-110 computer with real time Xenomai kernel Linux. Cheetah-cub does not have any sensors other than the internal low-level motor control sensors.

HyQ

HyQ (Hydraulically-powered Quadruped) is a versatile quadrupedal robot designed at the Italian Institute of Technology [Semini et al., 2011]. It weights 90[Kg] and the dimensions are about $1 \times 1 \times 0.5$[m] (height/length/width). Each leg consists of a hip AA joint actuated by a DC brushless electric motor, hydraulically actuated hip PR and knee FE joints, and a passive prismatic lower leg. The legs have considerably large work spaces, and each joint's range of motion is about 120°. HyQ is equipped with joint position sensors (absolute and relative), joint torque sensors, IMU, foot spring compression sensors, stereo cameras, and sensors for hydraulic states. HyQ can be powered by either an onboard hydraulic system, or by an external one resulting in a roughly 20[Kg] lighter platform. The control is done on a PC104 with real time linux working at 1[KHz].

LittleDog

LittleDog [Murphy et al., 2010] is a small sized quadruped made by Boston Dynamics for DARPA's learning locomotion program [Pippine et al., 2011], for the purpose of exploring the fundamental relationships among learning, terrain complexity, and locomotion control. LittleDog weighs 2.85[Kg] and sizes about $0.34 \times 0.16 \times 0.18$[m] (height/length/width). Each leg has three electrically actuated degrees of freedom: hip AA, hip PR and knee FE, and a compliant prismatic lower leg. Each leg has a large range of motion which allows for climbing over obstacles as big as the robot's leg length. The robot is equipped with joint position sensors, an IMU, and two-axis strain gauges on the lower legs to sense the contact states. The low-level motor control is done onboard at 500[Hz], and the high-level control is done off-board, communicated through wireless communication.

RoboCat-1

RoboCat-1 [Kotaka et al., 2013], made in Toyota Technological Institute, has been designed to be a durable and low-cost robot for rough terrain locomotion tests. Its weight is 6.8[Kg] and sizes about $0.23 \times 0.34 \times 0.30$[m] (height/length/width). Each leg has two actuated degrees of freedom, hip PR and knee FE, so the legs are only working in the Sagittal plane. Motors are powered using Harmonic Drive Systems FHA-8C series AC servo motors, and are torque-controlled. The robot does not have any passive compliance element, and is stiff-by-nature. There are two single axis gyroscopes to measure trunk's roll and pitch. Each foot is equipped with one force sensing resistor measuring the ground reaction forces. The robot is operated using MATLAB xPC Target system, and the high-level control is done at 1[KHz] off-board.

StarlETH

StarlETH [Hutter et al., 2012], made in the Autonomous Systems Lab at ETHZ, is a medium-sized quadrupedal robot made to be fast, efficient and versatile. StarlETH weighs about 23[Kg] and the dimensions are about $0.33 \times 0.51 \times 0.37$[m] (height/length/width). Each leg has three active degrees of freedom namely hip AA, hip PR, and knee FE. Legs are two segmented with equal segment lengths. The range of motion allows for the elbows and knees to point toward each other, away from each other, or both pointing forward or backwards. All the joint are quipped with series elastic actuators using pre-compressed linear springs. Maxon EC-4pole 200[W] motors are used for all the actuated joints. All joints are equipped with sensors to measure joint angles and spring deflections. Differential pressure sensors (Freescale Semiconductor Silicon Pressur eSensor MPXV5004) with air-filled balls are used for contact sensing. The trunk includes a Xsens MTi IMU. A centralized host-PC is used for high-level control. Low-level motor controllers (Maxon EPOS2 70/10) are used for the control of individual actuators, and the communication to the high-level PC is done through CAN bus. The high-level control can be realized at 400[Hz].

Tekken2

Tekken2 [Kimura et al., 2007a] is the successor of Tekken [Fukuoka et al., 2003], made in Kyoto Institute of Technology. It has the same design as Tekken (described below), but in comparison has on-board power and is about 25% larger. The weight is 4.3[Kg] including batteries, and the dimensions are $0.25 \times 0.30 \times 0.14$[m] (height/length/width). Each leg includes three actuated joints for hip yaw (not hip AA), hip PR, and knee FE, and a passive ankle with a spring-lock mechanism. Actuation is done using DC motors, 23[W] for hip PR and knee FE, and 8[W] for hip yaw. The robot is equipped with two rate gyros and two inclinometers in order to calculate body's roll and pitch. All joint are equipped with encoders. The ankle joint includes urethane gel which is compressed elastically in the stance phase, thus the ankle encoder can be used to detect stance, swing, and even stumbling (through the passive retraction of the foot). The robot utilizes an on-board PC with RT-Linux and TITECH-wire IO boards.

2.3.3 Control Methodologies

In this section we discuss the control methodologies used on the aforementioned robots. We do not provide an exhaustive list of all the control methods implemented on the robots, and rather focus on the ones which has been used to demonstrate rough terrain locomotion.

BigDog

BigDog's control is built on top of the knowledge acquired from experimenting with the classic Raibert's control [Raibert et al., 1986]. Raibert's control divides the dynamic locomotion control to three rules: 1) support the body by vertical hopping; 2) adjust the body attitude by

Robot	Weight [Kg]	Dimensions [m]	Type	Low-level	High-level
BigDog	90	1 × 1 × 0.3	Visually-guided + Blind	Hydraulic, Torque	1) Nominal control by Raibert's method; 2) Adjustments of GRFs; 3) Reflex responses to external disturbances; 4) Posture control by foot placement and kinematic adjustments.
BISAM	23	0.7×?×?*	Blind	Electric, Position	1) Matsuoka oscillators for gait timings; 2) Matsuoka oscillators to generate joint positions; 3) swing/stance timing adjustments depending on contact; 4) learning of "sensor - ankle" and "sensor - overriding postural commands" functions.
Cheetah-cub	1.1	0.16 × 0.2 × 0.1	(Limited) Blind	Electric, Position	1) Parametric open-loop CPGs implemented as phase oscillators with output shaping.
HyQ	70 – 90	1 × 1 × 0.5	Visually-guided + Blind	Hydraulic + Electric, Torque	1) Task-space CPG; 2) Kinematic foot adjustment; 3) Floating-based inverse dynamics; 4) null-space trunk control; 5) capture point push recovery; 6) stereo-vision to modulate control references.
LittleDog	2.85	0.34 × 0.16 × 0.18	Visually-guided + Blind	Electric, Position + Torque	1) Foothold planning; 2) COG trajectory generation; 3) Floating-based projected inverse dynamics with PD and force feedback.
RoboCat-1	6.8	0.23 × 0.34 × 0.30	Blind	Electric, Position + Torque	1) Constrained COG trajectories; 2) Foot trajectories with inverse kinematics; 3) Friction and gravity compensation; 4) Active compliance control based on contact force deviation; 5) Angular momentum control affecting reference trunk orientation.
StarlETH	25	0.33 × 0.51 × 0.37	Blind	Position + Torque	1) State estimation; 2) Gait graph-based behavior generation with step placement; 3) VMC or operational-space control to generate joint torques;
Tekken2	4.3	0.25 × 0.3 × 0.14	Blind	Electric, Position	1) Matuoka oscillators for phase generation; 2) Virtual spring-dampers with event-based gain switching; 3) Reflexes and responses to keep balance and correct the stepping.

Table 2.2: Overview of rough terrain locomotion quadrupeds. We report standing hip height as the height, shoulder to hip distance as the length, and contralateral shoulder separation as the width. (*) The exact dimensions of BISAM could not be found in the literature.

modifying the hip torques in the stance phase; 3) exploit foot placement to control the speed and the kinetic energy of the system. However Raibert's control does not address the problem of rough terrain locomotion [Raibert et al., 2008], and is rather designed for dynamic and fast locomotion on flat terrain.

The control methodology implemented on the BigDog extends the classic Raibert's control by adding modules for rough terrain locomotion. The exact details of the control is not publicly undisclosed, however partial information can be extracted from the published material [Playter et al., 2006, Raibert et al., 2008, Wooden et al., 2010]. The main high-level control elements can be listed as following:

- The stereo vision system is used to make a 3D map of the terrain in front of the robot and to find a clear path. BigDog can also feel its way along the rough terrain (using the

leg sensors) if exteroception is not available;
- Estimates of lateral velocity and acceleration are used for lateral stability;
- The controller regulates the ground reaction forces and tries to distribute the load equally on the legs. Moreover, the controller tries to direct the ground reaction force vectors towards the hips;
- The body attitude and height is adjusted respective to the local terrain;
- The positions of the feet are adjusted to compensate for body orientation, e.g. placing the feet more forward while descending a slope;
- At the individual leg level, virtual model control (VMC) [Pratt et al., 2001] is used to generate torque commands to follow a desired foot trajectory plan.

BISAM

The high-level control of BISAM [Ilg et al., 1999] is based on the idea of using neural oscillators to generate coordinated locomotion rhythms. At the leg level, superimposition of two Matsuoka oscillators is used to generate periodic rhythms for each joint, and each Matsuoka oscillator consists of three neurons. One oscillator is used for generating the swing trajectory, and one for the stance. Use of two separate oscillators allows for manipulation of each phase's duration. For example, swing time can be extended if the contact is missing (hole in the ground). This is implemented through an additive feedback from the binary contact information to the membrane potential. For the interlimb coordination, a separate four neuron oscillator is used to produce timing signals for the joint oscillators.

At an intermediate level, a reinforcement learning procedure is used to learn offset profiles which are added to the oscillator outputs. This is used to learn a function from the force and joint angle sensing to the output offsets. The function is modeled as a radial basis function network (RBF).

Finally, at the highest control level, posture control can dominate the oscillator output. Posture control consists of stability control and center of gravity control. Both are implemented using reinforcement learning on a fuzzy set of predefined actions like shifting the body to one side or lowering a leg. The learning uses the foot sensors, inclinometers, and gyros information to generate the activation level of the predefined actions.

CheetahCub

Cheetah-cub does not have any sensors (except for internal motor encoders). The idea behind the Cheetah-cub is to have smart mechanics which, accompanied by simple open-loop control, allows for robust locomotion. The control of the Cheetah-cub is done using CPGs. A network of phase oscillators with output shaping is used to generate coordinated sine-wave profiles for the hip PR, and smooth profiles with two peaks for the leg FE (one for swing foot clearance and one for active leg shortening in stance). A control duty factor parameter is also introduced to

modify the respective duration of swing and stance as needed. A few hundred of experiments are done to find the optimal parameters for dynamic and stable gaits. With the tuned CPG, the robot can locomote on flat ground with a maximum speed of about 1.4[m/s], about 7[BL/s]. The robot can also go over step-downs of about 20% of its leg length, and recover from small lateral perturbations thanks to its leg design. However the rough terrain capabilities of the robot is quite limited as the control is open-loop.

HyQ

HyQ has been used for both visually-guided and blind rough terrain locomotion. [Barasuol et al., 2013] presents a rather complete control framework for blind rough terrain locomotion. Their control exploits the idea of using a horizontal frame as the reference for generating foot trajectories. The horizontal frame is a floating frame located at the shoulder/hip, pointing forward, and parallel to the ground plane. The control method consists of:

- Task-space CPG to generate foot trajectory in the horizontal frame. In doing so, the foot trajectory is decoupled from trunk orientation. Foot trajectories are modeled as cut ellipsoids;
- Kinematic adjustments to the CPG output based on the body inclination to keep the foot trajectory close to the ground and prevent weak contact;
- Floating-based inverse dynamics with PD feedback to track the desired joint profiles, obtained by inverse kinematics of the desired foot trajectories;
- Torques to track a desired trunk orientation in the null-space of the foot trajectory task.
- Push recovery by calculation of the instantaneous capture point. The needed trunk velocities are obtained through state estimation.

HyQ can also observe the rough terrain using a stereo camera. [Havoutis et al., 2013, Bazeille et al., 2013] use the blind rough terrain locomotion framework of [Barasuol et al., 2013] as basis, and modulate its parameters to locomote over perceived terrain. Based on the sensed terrain, step height is modulated proportional to obstacle height, forward velocity and duty factor are reduced with respect to terrain difficulty, and the gait is changed to a statically stable walking gait if the terrain is too difficult.

LittleDog

LittleDog has been mostly used to develop control for visually-guided rough terrain locomotion. As a matter of fact, in the DARPA's learning locomotion program, the 3D map of the terrain was given, and robot's position and orientation states were given online through a motion capture setup. This means that the control in this case directly involves planning. There are several papers which discuss the case of visually-guided rough terrain locomotion with the LittleDog robot, including [Kalakrishnan et al., 2009, Shkolnik et al., 2010, Kolter and Ng, 2011, Neuhaus et al., 2011], and discussing the details of all these approaches is out of the

scope of this thesis. Here we only discuss [Buchli et al., 2009], where the control is designed to deal with unperceived terrain and obstacles.

The essence of the control approach presented in [Buchli et al., 2009] is floating-based inverse dynamics control. Inverse dynamics control creates the possibility to be actively compliant if the feedback loop is chosen to be with low gain. The authors address that the constraint Jacobian in the equations of motion can be decomposed utilizing QR decomposition. This has the benefit of making the feedforward torque generation independent of contact forces. Nevertheless, contact forces can be estimated using the inverse dynamics, and force control terms are written to counteract the feedback PD controller in case of unexpected contact.

Inverse dynamics need joint trajectories which are twice differentiable. To generate such trajectories, first a foothold selection algorithm is deployed, and then the COG (Center Of Gravity) trajectory is designed to satisfy the differentiability and kinematic constraints. Swing leg trajectories are constructed using splines taking into account the convex hulls of known obstacles.

[Buchli et al., 2009] demonstrates that the use of floating-based inverse dynamics with PD and force feedback signals enables the robot to blindly locomote over rough terrain with obstacles as big as 30% of the leg length, with a static walking gait of about 0.3[BL/s] forward speed.

RoboCat-1

RoboCat-1 is able to perform blind rough terrain hopping and in-place trotting [Ugurlu et al., 2013]. The control of RoboCat-1 is divided into high-level and low-level controllers. The high-level control consists of:

- COM trajectories which are defined as 6th order polynomials satisfying similar take-off and touch-down velocities and accelerations while respecting the joint limits;
- Foot trajectories generated using polynomials;
- Inverse kinematics using the reference trunk orientation, COM and foot trajectories to generate the joint positions respective to the world frame task-space values.

The generated high-level position commands are then fed to low-level controller, which has the following elements:

- Friction compensation utilizing friction hysteresis identification, and gravity, Coriolis and centrifugal compensation;
- Active compliance control by generating compensating joint position feedback based on the feet force measurement error;
- Angular momentum control calculating the angular momentum rate change of the main body and updating the reference trunk orientation.

StarlETH

The control of StartlETH, as presented in [Hutter et al.], includes three parts: 1) Robot state estimation fusing the data from different sensors to estimate task space positions and velocities as well as trunk orientation [Bloesch et al., 2013]; 2) High-level whole-body controller (more below); and 3) low-level joint controller utilizing torque-control in the stance phase and position-control in the swing phase.

The high-level control is responsible for behavior generation and control. At the behavior generation level, the robot uses static walking with planned footholds for rough terrain locomotion. Dynamic trotting is used for faster locomotion, where stepping location is calculated based on the desired speed, and to keep balance. Stepping timings are derived from desired gait graphs. Body roll and pitch are kept constant. For other dynamic gaits like bounding, the vertical movement of the body is superimposed. After that, the behavior control is done using either Virtual Model Control (VMC), or using hierarchical task-space inverse dynamics (operational-space control). For the operational-space control method, task-space tasks are hierarchically prioritized as a least-squares problem and numerical optimizations are used to solve for joint torques. For the task of locomotion, the task priorities are ensuring support constraint, controlling trunk height, robot attitude, main body turning rate, and main body velocity. The advantage of using the hierarchical task-space inverse dynamics is that different tasks/constraints in different coordinate systems can be put together in the same control optimization formulation. This includes optimization of contact forces or torque distributions.

Tekken2

The control methodology used on Tekken (and Tekken2) is a successful example of how closed-loop CPGs can be used for rough terrain locomotion, but is a rather complex way to implement locomotion control. The control on Tekken aims to satisfy the necessary conditions for stable dynamic walking on irregular terrain [Fukuoka et al., 2003], which includes:

- Keeping the stride period short to maintain high WSM (Wide Stability Margin);
- Keeping the angular momentum constant at the moment legs land or leave the ground;
- Maintaining the phase difference between body roll and leg pitch, and the phase difference between the legs themselves.

The control on Tekken is divided into three main parts: rhythm generation by CPGs, virtual spring-damper joint control, and reflexes. The CPG model is based on Matsuoka oscillators. For each leg, a neural oscillator consisting of two mutually inhibiting flexor and extensor neurons is used. The output of CPG is the phase signal for each leg, which is calculated from superimposition of the flexor and extensor output signals. Each neural oscillator is also affected by an additive feedback term $k(\theta - \theta_0)$, with k being the feedback gain, θ the actual hip angle, and θ_0 the origin of hip joint when standing still. Authors call this the tonic stretch response.

Virtual spring-dampers are used along the CPG phase signals to generate the joint angle profiles. The leg motion is divided to three stages, namely swinging up, swinging forward, and pulling down/back. For each of these phases a set of reference leg postures are defined, and relatively low stiffness virtual spring-dampers (PD controller) are used. The PD gains are different for each stage.

Finally, reflexes/responses are added to help the locomotion control and to respect the necessary conditions defined above. Reflexes are direct torque feedforward signals, while responses are feedback signals to the CPG:

- Flexor reflex to prevent stumbling in the swing phase;
- Sideways stepping reflex using the hip yaw to stabilize the body roll on lateral slopes;
- Vestibulospinal reflex/response which corrects body pitch by flexing or extending the legs (e.g. front legs flexed on an upwards slope);
- Tonic labyrinthine response, which is similar to the vestibulospinal response, only for the body roll;
- Corrective stepping reflex to react to a missing contact by increasing the input to the leg extensor neuron;
- Crossed flexor reflex to prevent stumbling when the robot is going over a step down.

2.3.4 Summary

We presented a diversity of robots which are used to demonstrate rough terrain locomotion. This includes robots ranging from 1 to 90[Kg] in weight, being position- or torque-controlled, and having few to many different sensors.

From the review presented, it seems that big machines mostly use torque-control while smaller ones use position-control at the low-level. One reason can be that bigger machines are typically much more powerful, and can be dangerous if they are high-gain position controlled. Use of a torque-control method like inverse dynamics helps having active compliance, which, if implemented correctly, makes these robots safer and softer during physical interaction. As for the cheap and small machines, the inclusion of torque-controlled motors is not straightforward. This is because off-the-shelf torque controlled motors are quite expensive if they are small and powerful (or they should be custom made), or have big motor boards which does not fit into smaller robots. Moreover, one idea behind building smaller robots is the budget limitation, and accurate torque sensors are not cheap. It is also worth mentioning that, to exploit the features of a torque-controlled robot, controllers like inverse dynamics should be used, which are computationally heavy for smaller robots with weaker (typically single-core) on-board computers.

Different control methods are used on different platforms including classic and bio-inspired approaches, and have been able to demonstrate successful locomotion over rough terrain. It can be observed that as the terrain gets more and more difficult, the gaits used become slower,

and tend to be static rather than dynamic. Additionally, more difficult setups demand force control or active compliance.

From a gait generation perspective, one can roughly summarize the presented controllers in three categories (and sometimes a mixture of them):

- Controllers which generate joint position commands for a low-level PID controller, and feedback mechanisms like posture control modify the joint position references;
- Controllers which provide joint positions, velocities, and accelerations for an inverse-dynamics controller generating joint torques, and posture control is performed by additional torques;
- Controllers which perform the control in the task-space utilizing concepts like virtual model control, with a phase-generator or a state-machine to provide the stepping sequence.

The first method is simple but does not result in active compliance (other than when very small gear ratios are used). The second method brings joint-space active compliance, if used properly. The third method offers task-space active compliance. The second and third methods need torque-control capability.

2.4 Conclusion

Since this thesis focuses on the design of pattern generators for (blind) rough terrain locomotion, we reviewed different methodologies to construct computational CPG models, as well as different control methodologies applied to the problem of rough terrain locomotion. Based on the material presented, following is a list of desired features for successful control over rough terrain locomotion when exteroception is unavailable:

1. At the low-level, torque-control capability and floating-based inverse dynamics. Examples of LittleDog, HyQ and StarlETH reveal how these features lead to safe and compliant interaction with the environment;
2. Feedforward pattern generation for forward locomotion. Of course the nominal patterns of locomotion can also be generated in a feedback-driven manner, but having an exogenous pattern generator eliminates dependency on the sensory information to do simple flat terrain locomotion, which can be a basis for more sophisticated control. Keep in mind that with the inclusion of smart mechanics, like in Cheetah-cub, open-loop CPGs are good enough for flat ground locomotion, and simple rough terrain scenarios;
3. Kinematic model-based posture control as opposed to heuristics. Control on Tekken2 is an example of how posture control can be done, though successful, in a heuristic and indirect way;
4. A few selected reflexes. Although the whole posture control should be implemented systematically and model-based, a few reflexes like stumbling correction reflex seem to be necessary;

5. Adjustments of stepping location to keep balance. Foot placement is needed when the perturbations are big, and modified steps should be taken to prevent the body from tipping over.

As we will see in the next chapters, we will propose a control methodology which includes (2.), (3.), (4.), and partially (5.). Due to the capabilities of the robots that we use, we are unable to include (1.) as it needs torque-control capability.

3 Problem Statement

آن کس که نداند و بداند که نداند لنگان خرک خویش به منزل برساند

The one who does not know, and knows that he does not know,
will eventually, with whatever the difficulties, reach his goal.

Ibn-e-Yamin

This thesis addresses the problem of designing a modular controller for blind rough terrain locomotion in a systematic way. The aim is to design the control at the motion generation level, and leave out the low-level motor control. In this chapter we will define the control problem, its properties, and the constraints on the control and the sensory information available. The benchmarks used to quantify the results will be briefly described. We will then illustrate the big picture of our proposed control methodology to solve the stated problem. This chapter is accompanied by the list of symbols and abbreviations used throughout this thesis.

3.1 Formal Problem Statement

The core of the control problem that this thesis will address is traversing over rough terrain in limited time while keeping the trunk angles contained. The terrain geometry is going to be unknown, including the type of the terrain. The control problem is formalized as follows:

3.1.1 Control Problem

Objective

Given:
Robot R
Unknown terrain $\Gamma(\boldsymbol{\rho})$, with $\boldsymbol{\rho}$ describing environment's geometry
Known robot initial state $\mathbf{s}_0$
provide:
Control modules $\Pi_i(\boldsymbol{\alpha}_i)$, with $\boldsymbol{\alpha}_i$ being the parameters of the i-th module
such that:
R traverses over $\Gamma(\boldsymbol{\rho})$ using $\bigcup_i \Pi_i(\boldsymbol{\alpha}_i)$
subject to:
$t_{final} < t_{max}$
$\forall t, \psi_{roll}(t) < \psi_{roll,max}$
$\forall t, \psi_{pitch}(t) < \psi_{pitch,max}$

3.1.2 Properties

The control problem described above is accompanied by the following properties:

Environment A terrain $\Gamma(\boldsymbol{\rho})$ which is geometrically rough, including randomized height maps and inclined surfaces. We do not specifically address terrains which the roughness is due to moving parts or excessively low or high friction (e.g. ice and mud). Nevertheless, we test our controllers on a subset of these environments.

Sensing Only proprioception is available. No external information about the terrain is given, including the type of the terrain, friction constants, etc.

Modularity The control should be modular and each module should have a tangible meaning. Each module should be testable on top of lower level modules in a hierarchical manner.

3.1.3 Constraints

There are several constraints on the controller design which are due to the robotic platform that we use (Oncilla), the available equipment budget, and the AMARSi project's needs. The list of constraints is as follows:

Low-level control The low-level controller is a position-controller. Nevertheless, we explore simple alternatives if torque-control is available.

Sensing Available sensing consists of joint angle sensing (encoders), body rotation sensing (high-end IMU), and on-off contact sensing (contact sensors). No continuous 3D or 6D force sensing is available. No joint torque sensing is available.

Computational burden There is limited computational power available. The control should be as computationally heavy as needed and not more. The control should be implementable on a single-core PC satisfying real-time constraints.

3.1.4 Benchmarks

A collection of six different benchmarks are used to quantify the capabilities of our control methodology (Figure 3.1). We may use a mixture of different benchmarks for demonstration purposes. Not all of the benchmarks are used for all of the platforms. The benchmarks are:

Uneven terrains A terrain consisting of a randomized height map. The difficulty of the terrain is quantized by two factors: 1) the maximum height variation of the map compared to the robot's leg length; 2) The maximum local inclination. An indoors rough terrain setup is constructed for the hardware experiments.

Rocky setups Similar to the uneven terrain, but made from small rocks and pebbles. The terrain difficulty is assessed by the maximum height variation compared to the leg length.

Inclined surfaces Upwards or downwards inclined surfaces. The difficulty is determined by the degree of inclination.

Steps A vertical step (upwards or downwards) placed in a random distance. The difficulty is determined by the height of the step compared to the leg length.

External pushes 3D external perturbations to the main body of the robot. The size and the direction of the perturbations are not known in-advance. The difficulty of this benchmark is measured by the magnitude and duration of the external force.

Asymmetric load carriage asymmetric load placed on one side of the robot. The difficulty is assessed by the weight of the load and its displacement.

3.2 The Big Picture

Taking into account the goals and constraints specified above, we need to design a modular controller which is computationally light, does not explicitly depend on torque-control capability, can work with limited sensory information, and enables the robot to blindly locomote over rough terrain.

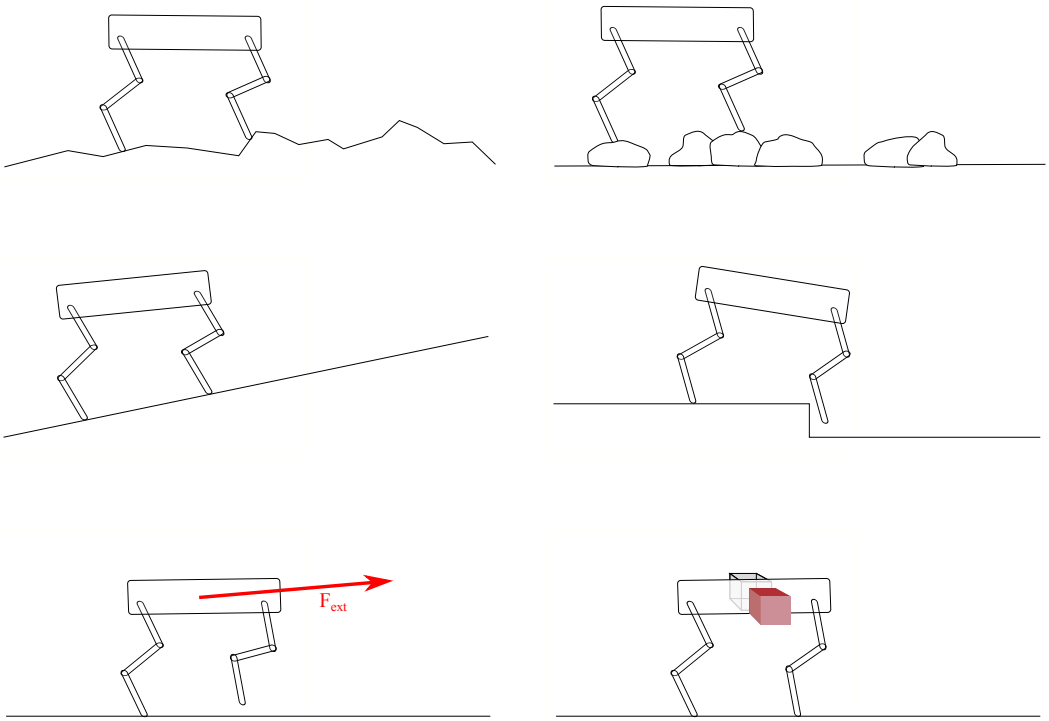


Figure 3.1: Different benchmarks used to evaluate the performance of the control methodology. From top-left to right-bottom: uneven terrain, rocky setup, inclined surface, step-down, external push, and asymmetric load carriage.

Our proposed control architecture, explained in details in Chapter 6, exploits the concept of Central Pattern Generators at the bottom of the hierarchy (Figure 3.2). CPGs, if designed properly, can be computationally cheap [Ijspeert and Crespi, 2007], generate fast gaits [Spröwitz et al., 2013], and provide the basis for feedback integration [Fukuoka et al., 2003]. Moreover, the operation of the CPG can be independent from the sensory input, which keeps the controller working even if no sensor is available (of course with a poorer quality of task execution). This is different from a fully state-feedback driven system. Chapter 4 will explain how we can systematically design nonlinear oscillators used as CPGs.

With the available information from encoders, IMU, and on/off contact sensors we can know which legs are on the ground, and how the robot is rotated in space. Based on these information, a kinematic posture controller is designed, as detailed in Chapter 5. The posture controller provides feedback signals for the CPG to keep the body upright while locomoting. Moreover, using the contact information along the CPG's timing signals, the controller knows if undesired (or missing) contacts are happening. We add momentary and quick reflexes to account for such situations, as explained also in Chapter 5.

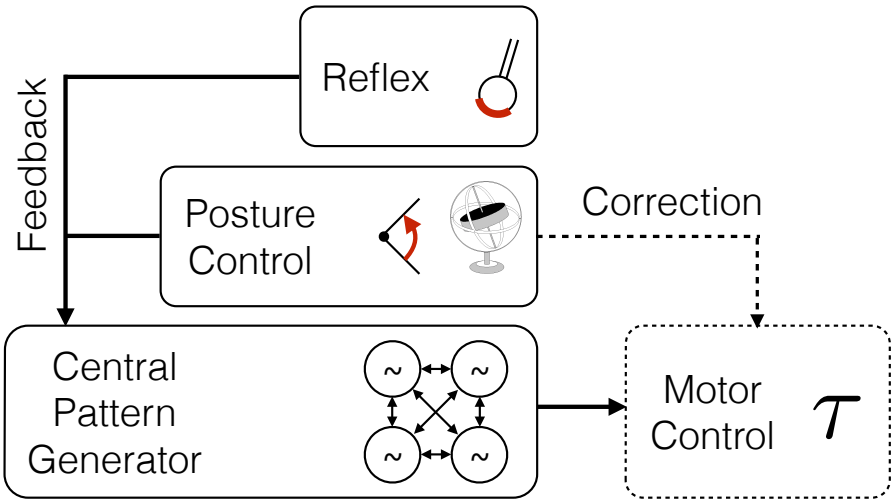


Figure 3.2: The essence of the modular control architecture proposed in this thesis. A Central Pattern Generator (CPG) model resides at the bottom of the hierarchy, and exogenously generates control inputs. Posture controller, based on vestibular and proprioceptive sensing, generates feedback signals for the CPG. Finally reflexes are generated (mostly) based on the contact information, and are given to the CPG. Optionally, if there is access to the motor control module, correction signals are generated at the motor control level. This figure only depicts the gist of the proposed control architecture, and a detailed version is provided later in Chapter 6.

It is worth mentioning that, in all the process of designing the control modules, being systematic is our quality standard. We would like to avoid heuristic as much as possible. Nevertheless, intuition is applied minimally.

3.3 List of Types, Symbols, and Abbreviations

Following is the generic list of symbols used throughout this thesis. Please note that a subscript or superscript will not change the type of the symbol.

Kind	Description	Example
Non-bold lowercase	Scalar	$a, b_x, \alpha, \beta_i, \ldots$
Bold lowercase	Column vector	$\mathbf{v}, \mathbf{z}_i, \boldsymbol{\theta}$
Bold uppercase	Matrix	$\mathbf{A}, \mathbf{C}_i, \boldsymbol{\Xi}$
Blackboard bold uppercase	Vector subspace	$\mathbb{R}, \mathbb{B}$
Calligraphic uppercase	Set of equations	$\mathscr{F}, \mathscr{D}_{\mathbb{B},r}$

Table 3.1: Table of types

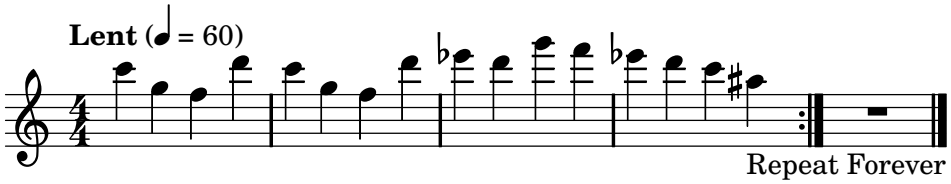
Symbol	Usage
$\boldsymbol{\theta}, \theta_i$	Phase in (extended) polar coordinates
$\mathbf{r}, r_i$	Radius in (extended) polar coordinates
$\boldsymbol{\Phi}, \phi_{ij}$	Phase lag
c, c_{ij}	Coupling strength
$\mathbf{f}(.), f_i(.)$	Oscillator shaping function
$\mathscr{D}_{\mathbb{A},x}(.)$	Dynamical system in vector subspace $\mathbb{A}$ for variable x
ϑ	Oscillation frequency, locomotion stride frequency
γ	Oscillator convergence rate
$\boldsymbol{\xi}, \xi_i$	Angular velocity feedback
$\boldsymbol{\tau}, \tau_i$	Commanded joint torque
$\boldsymbol{\delta}, \delta_l$	On/off sensed contact state
$\mathbf{q}, q_i$	Sensed joint angle
$\mathbf{R}$	Trunk rotation matrix
$\mathbf{R}_{str}$	Trunk rotation matrix excluding yaw
$\boldsymbol{\psi}, \psi_{\{roll,yaw,pitch\}}$	Euler angles representing robot's trunk orientation represented as roll (around x axis), yaw (around y axis) and pitch (around z axis)
$\boldsymbol{\psi}_{gnd}, \psi_{\{roll,yaw,pitch\},gnd}$	Euler angles representing local ground inclination
$\mathbf{J}_l$	Jacobian of forward kinematic for the tip position of lth leg w.r.t. world
$\mathscr{R}(.)$	Function to construct rotation matrix from Euler angles
$\mathscr{F}(.)$	Forward kinematics

Table 3.2: Table of symbols

Abbreviation	Meaning
AA	Abduction/Adduction degree of freedom of shoulders or hips
PR	Protraction/Retraction degree of freedom of shoulders or hips
FE	Flexion/Extension degree of freedom of elbows or knees
NK	Flexion/Extension degree of freedom of wrists or ankles
BL	Body length defined as shoulder to hip distance
LL	Leg length defined as the standing hip height
FT	Foot trajectory

Table 3.3: Table of abbreviations

Methodology Part II

4 Morphed Oscillators

Simplified from "Dearly Beloved"
Kingdom Hearts, Yoko Shimomura

Publication

Most of the material presented in this chapter is taken from:

– Mostafa Ajallooeian, Jesse van den Kieboom, Albert Mukovskiy, Martin A. Giese and Auke J. Ijspeert. A general family of morphed nonlinear phase oscillators with arbitrary limit cycle shape. *Physica D: Nonlinear Phenomena*, 263:41–56, 2013.

As discussed in the first part of this thesis, we aim to design a modular control architecture which, at a high level of abstraction, represents the locomotion control circuitry, and enables a robot to locomote over rough terrain. At the core of such an architecture resides the rhythm generation mechanism, known as the Central Pattern Generator (CPG). Looking at the CPGs from a high-level abstraction viewpoint, they can be considered to be nonlinear oscillators generating activation rhythms with arbitrary signal shapes, and accepting feedback from other parts of the locomotion circuitry.

This chapter addresses the problem of designing nonlinear oscillators with arbitrary limit cycle shapes. We present how these kind of oscillators can be constructed in a *systematic* manner. What is presented in this chapter goes beyond the scope of the locomotion control problem, and deals with nonlinear oscillator design in an abstract mathematics fashion. We

will see in the next chapters that even the simple cases of the oscillators that we present in this chapter can be enough to implement a CPG for locomotion control, and this chapter is presented as it is for the sake of theoretical completeness.

4.1 Introduction

Nonlinear oscillators have been widely used in different fields ranging from abstract to applied sciences and engineering [Khalil and Grizzle, 2002]. Their capability of entrainment, synchronization and smooth modulation of their output signal makes them appropriate tools for modeling natural phenomena and for control [Khalil and Grizzle, 2002, Kovacic and Brennan, 2011, Matsuoka, 1987].

One challenge in the field of nonlinear oscillators is how to design oscillators with desired limit cycle shapes [Ijspeert, 2008]. One key application of such oscillators is in system and controller design. The benefit of using nonlinear oscillators in controller design is that the control references can be coded into dynamical systems, and if done properly, properties like smoothness, continuity and stability after state perturbation are preserved. Now if the desired control references are functions of desired shapes or structures, then there is a need for oscillators capable of having arbitrary limit cycle shapes.

We take inspiration from the works mentioned in Chapter 2, and present a general way to convert an existing phase oscillator to any desired nonlinear phase oscillator with well defined, controllable properties. We believe that our contribution is three fold: 1) We present a *general*, *simple* and *systematic* way to design nonlinear phase oscillators. As a consequence, we obtain not a single, but a family of nonlinear phase oscillators exhibiting desired limit cycle shapes. This gives researchers the ability to easily design custom nonlinear phase oscillators. 2) The general methodology we present here can be taken as a unifying view on well known techniques including PD control and rhythmic Dynamical Movement Primitives. This enables the comparison of these methods under a same conceptual framework. 3) The mathematical formulation gives room to introduce a simple nonlinear oscillator which can have a *custom convergence behavior*, independent of the other properties like arbitrary limit cycle shape. This is a useful property providing the possibility to define the way to converge to the limit cycle, which is potentially useful for initiation in many cyclic motion control problems.

The rest of this chapter is organized as follows: The design methodology is explained in Section 4.2. Example realizations of the introduced methodology is given in Section 4.3. Extension to multi-dimensions is explained in Section 4.4. We then analyze the stability conditions in Section 4.5. Section 4.6 details how arbitrary convergence behavior can be obtained. Section 4.7 discusses building nonlinear phase oscillators from data. Section 4.8 describes the applications of the morphed oscillators, and we summarize and discuss our work in Section 4.9.

4.2 Methodology

There are many nonlinear oscillators with different characteristics, including Hopf oscillator [Khalil and Grizzle, 2002], van der Pol oscillator [Khalil and Grizzle, 2002], Fitzhugh-Nagumo oscillator [FitzHugh, 1961], and many more, and each of these oscillators exhibit a different limit cycle shape. However, it is not trivial to design an oscillator with a *desired* limit cycle shape. We propose a systematic way to design phase oscillators (oscillators with an explicit phase variable, such as a Hopf oscillator expressed in polar coordinates, for instance) with arbitrary limit cycle shape.

One can take a simple phase oscillator, and try to morph this oscillator's limit cycle to obtain a desired one. So a mapping which takes the states of this oscillator and modifies them such that the outcome has the desired property is needed. This mapping can be arbitrarily complex, but in case of the phase oscillators, we show that scaling the radial state depending on the phase value is sufficient to modify the limit cycle shape.

Our methodology consists of an oscillator called the *base oscillator* (in phase plane $\mathbb{B}$, real plane $\mathbb{R}^2$), a *desired oscillator* (in phase plane $\mathbb{S}$, real plane $\mathbb{R}^2$) and a mapping between them (M). We aim to find invertible M such that any path in $\mathbb{B}$ is mapped to a unique path in $\mathbb{S}$ and vice versa. Consequently, with the correct choice of M, the limit cycle of the base oscillator in $\mathbb{B}$ will be mapped to the desired one in $\mathbb{S}$.

Let us assume that the desired limit cycle is defined by the mapping $\Xi_{\mathbb{S}} : \theta_{\mathbb{S}} \rightarrow r_{\mathbb{S}}$, with $\theta_{\mathbb{S}}$ and $r_{\mathbb{S}}$ respectively being the angle and radius in polar coordinates. Also assume a base oscillator for which its limit cycle can be defined in polar coordinates by the mapping $\Xi_{\mathbb{B}} : \theta_{\mathbb{B}} \rightarrow r_{\mathbb{B}}$. Now if the oscillator in $\mathbb{S}$ has a phase always equal to the base oscillator's phase, i.e. $\theta_{\mathbb{S}} = \theta_{\mathbb{B}} = \theta$, then the radius of the limit cycle of the oscillator in $\mathbb{S}$ can be defined with a *phase-based scaling* (also see Figure 4.1) :[1]

$$r_{\mathbb{S}} = f(\theta) r_{\mathbb{B}} \tag{4.1}$$

where f is a scaling function that scales based on the phase value. So a state $\{\theta, r_{\mathbb{S}}\}$ in $\mathbb{S}$ corresponds to a state $\{\theta, \frac{r_{\mathbb{S}}}{f(\theta)}\}$ in $\mathbb{B}$, and vice versa.

The mapping from space $\mathbb{B}$ to $\mathbb{S}$ can be written as $M(\theta, r) = (\theta, r f(\theta))$, and calculating the Jacobian determinant of M gives $|J| = f(\theta)$. So if $\forall \theta : f(\theta) \neq 0$ and f is C^1 differentiable, then M defines a C^1-diffeomorphism [Khalil and Grizzle, 2002] between $\mathbb{B}$ and $\mathbb{S}$. This means that each state in $\mathbb{S}$ corresponds to a *unique* state in $\mathbb{B}$ and vice versa. Having the diffeomorphism property, an intuitive way to form the desired limit cycle system in $\mathbb{S}$ is: [2]:

[1] Here we address a multiplicative relation between $r_{\mathbb{S}}$, $f(\theta)$ and $r_{\mathbb{B}}$. In general this can be any invertible relation $r_{\mathbb{S}} = g(f(\theta), r_{\mathbb{B}})$, and we are writing a follow-up paper on this concept.

[2] From here to the end of this section is the process to obtain the morphed oscillators. If reader is seeking for the final formula, he/she can skip to the end of section 4.2.1 and refer to Table 4.1 for a summary and an example.

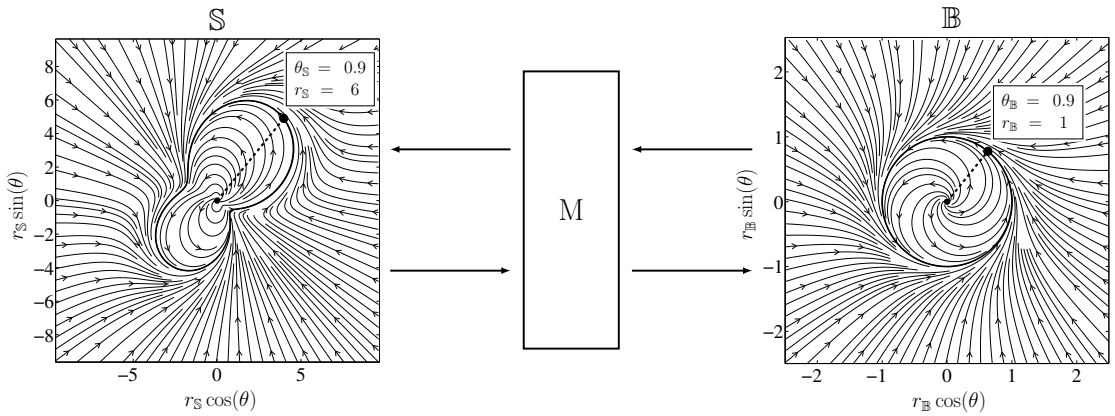


Figure 4.1: Example of a mapping between the desired oscillator (left) and the base oscillator (right). The main idea is to select an existing structurally stable oscillator, like the Hopf oscillator (right), and find a mapping (M) which shapes the limit cycle of this oscillator to a desired one (left). A phase-based scaling function $f(\theta)$ can implement M to map points from $\mathbb{B}$ to $\mathbb{S}$ and vice versa. For example, for the two points shown, the scaling function is $f(0.9) = \frac{6}{1}$.

1. Take the current state $\{\theta(t), r_{\mathbb{S}}(t)\}$ in $\mathbb{S}$
2. Map this state to $\mathbb{B}$ using $r_{\mathbb{B}}(t) = \frac{r_{\mathbb{S}}(t)}{f(\theta(t))}$
3. Use the dynamical system of the base oscillator in $\mathbb{B}$ to calculate $r_{\mathbb{B}}(t+\Delta t)$ and $\theta(t+\Delta t)$
4. Map $r_{\mathbb{B}}(t+\Delta t)$ back to $\mathbb{S}$ by $r_{\mathbb{S}}(t+\Delta t) = f(\theta(t+\Delta t))r_{\mathbb{B}}(t+\Delta t)$
5. Do $t \xleftarrow{update} t+\Delta t$ and continue from 1.

With the above algorithm we can mathematically write the process of the limit cycle generation as an iterative map, i.e. a discrete-time dynamical system (the continuous-time description will be given later):

$$\theta(t+\Delta t) = \theta(t) + \int_t^{t+\Delta t} \mathscr{D}_{\mathbb{B},\theta}(\theta(t))\,dt \tag{4.2}$$

$$r_{\mathbb{S}}(t+\Delta t) = f(\theta(t+\Delta t))\left(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))} + \int_t^{t+\Delta t} \mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))}\right)dt\right) \tag{4.3}$$

where $\mathscr{D}_{\mathbb{B},\{r,\theta\}}(.)$ are the differential equations of the base oscillator ($\dot{\theta} = \mathscr{D}_{\mathbb{B},\theta}(\theta)$, $\dot{r}_{\mathbb{B}} = \mathscr{D}_{\mathbb{B},r}(r_{\mathbb{B}})$). Since the base oscillator is a phase oscillator ($\dot{\theta} = \omega = \text{const}$) we have:

$$\mathscr{D}_{\mathbb{B},\theta}(.) = \omega = 2\pi\vartheta \tag{4.4}$$

where ϑ is the oscillator's frequency. Equations (4.2,4.3) introduce a general way to convert a chosen base oscillator to one with a desired limit cycle. The shape of the limit cycle in $\mathbb{S}$ is $\Xi_{\mathbb{S}}(\theta) = \Xi_{\mathbb{B}}(\theta)f(\theta)$.

4.2.1 Continuous-time Dynamical System

The oscillators obtained by Equations (4.2,4.3) are with discrete-time updates. A continuous-time dynamical system form can be obtained by calculating the limits of the forward differences of those equations when $\Delta t \to 0$. For θ we simply have (using Euler approximation for integration):

$$\dot{\theta}(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left(\theta(t) + \left(\int_{t}^{t+\Delta t} \mathscr{D}_{\mathbb{B},\theta}\left(\theta(t)\right) dt \right) - \theta(t) \right) = \mathscr{D}_{\mathbb{B},\theta}(\theta(t)) = \omega = 2\pi\vartheta \tag{4.5}$$

which is the definition of the phase dynamics of the base oscillator in $\mathbb{B}$. For $\dot{r}_{\mathbb{S}}(t)$ we have:

$$\dot{r_{\mathbb{S}}}(t) = \lim_{\Delta t \to 0} \left(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))} \frac{f(\theta(t+\Delta t)) - f(\theta(t))}{\Delta t} + \frac{f(\theta(t+\Delta t)\mathscr{D}_{\mathbb{B},r}(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))})\Delta t}{\Delta t} \right) \tag{4.6}$$

which gives:

$$\boxed{\dot{r_{\mathbb{S}}}(t) = \frac{r_{\mathbb{S}}(t)}{f(\theta(t))} \dot{f}(\theta(t)) + f(\theta(t))\mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))} \right)} \tag{4.7}$$

with $\dot{f}(\theta(t)) = \frac{f(\theta(t))}{d\theta(t)} \frac{d\theta(t)}{dt} = \omega f'(\theta(t))$. Equation (4.5) along with (4.7) gives a general way to obtain a desired continuous-time phase oscillator. The shape of the limit cycle can be arbitrarily defined by $\Xi_{\mathbb{S}}(\theta) = \Xi_{\mathbb{B}}(\theta) f(\theta)$ (either $f(\theta)$ is defined and then $\Xi_{\mathbb{S}}(\theta)$ is derived, or vice versa). The convergence behavior is defined by the dynamics of the base oscillator. We call the set of Equations (4.5,4.7) the *original form*.

The canonical evolution in Equation (4.7) is $\frac{r_{\mathbb{S}}(t)}{f(\theta(t))} \dot{f}(\theta(t))$, meaning that the amplitude of the oscillations generated by $\dot{f}(\theta)$ are magnified by $\frac{r_{\mathbb{S}}(t)}{f(\theta(t))}$ (Figure 4.2-top, dashed blue lines). So for large $r_{\mathbb{S}}(t)$ values, the amplitude of the oscillations will be largely magnified. If this effect is undesirable, then one can compensate for this effect by using the following radial differential equation:

$$\boxed{\dot{r_{\mathbb{S}}}(t) = \Xi_{\mathbb{B}}(\theta(t)) \dot{f}(\theta(t)) + f(\theta(t))\mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}(t)}{f(\theta(t))} \right)} \tag{4.8}$$

where $\Xi_{\mathbb{B}}(\theta(t))$ is the shape of the base oscillator's limit cycle. We call the set of Equations (4.5,4.8) the *compensated form*. State-time evolution and phase portraits of both original and compensated forms are depicted in Figure 4.2, for same desired limit cycle. As shown, the difference between these two forms is in their convergence behavior.

As Figure 4.2 depicts, for the compensated form, the $r_{\mathbb{S}}$ state can become negative (Figure

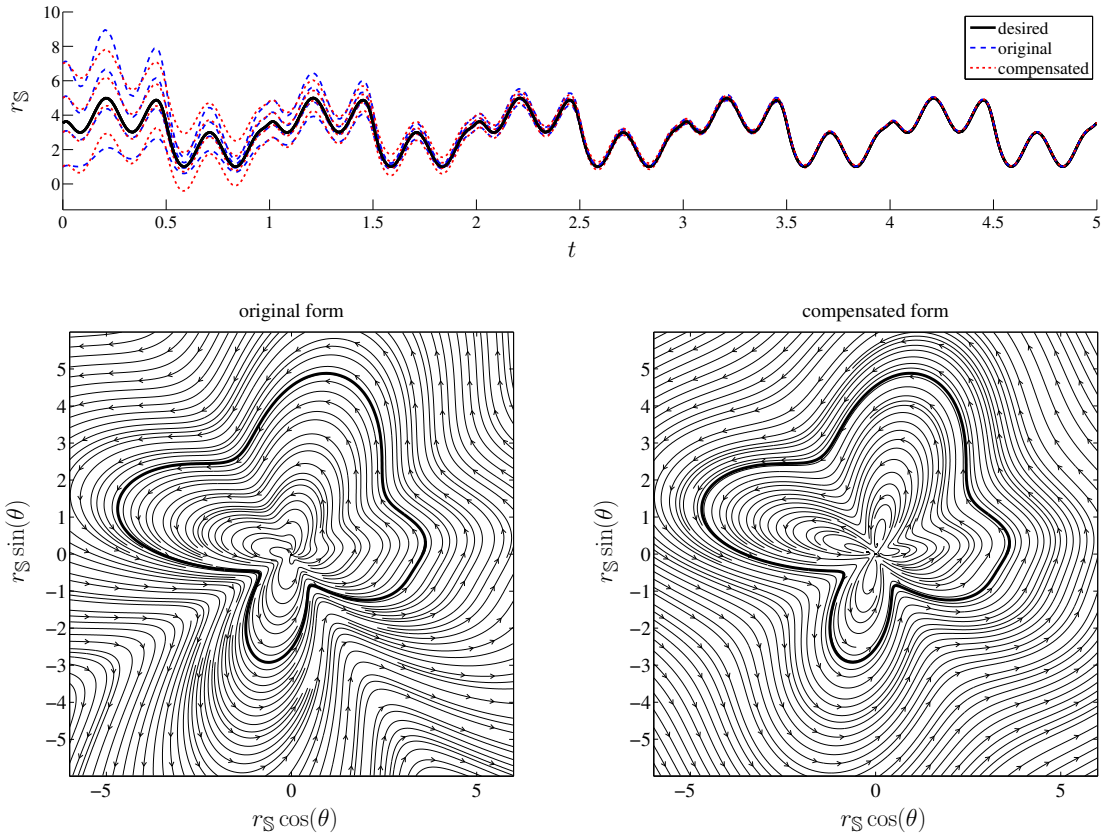


Figure 4.2: Comparison between the original and the compensated forms. The base oscillator used is an amplitude controlled oscillator with $\mu = 1$ and $\gamma = 1$ (please refer to Table 4.2 for the base oscillator's equations). The desired limit cycle is $\Xi_{\mathbb{S}}(\theta) = 3 + \tanh(5\sin(\theta)) + \cos(4\theta + 1)$. The contours of the original form are similar to the limit cycle and scaled by $\frac{r_{\mathbb{S}}(t)}{f(\theta(t))}$, while the contours of the compensated form become circular when they are away from the limit cycle. The phase portraits are only for $r_{\mathbb{S}} > 0$.

4.2-top, at $t \simeq 0.6$) even starting from positive initial values. This means that negative radius values are meaningful without shifting to the antiphase state. We term the space formed by $\{\theta, r_{\mathbb{S}}\}$ as *extended polar coordinates*, and discuss this more in Section 4.5 when analyzing stability conditions.

The obtained oscillator forms are first order dynamical systems. Positions $r_{\mathbb{S}}(t)$ and velocities $\dot{r}_{\mathbb{S}}(t)$ are continuous if not directly perturbed. If f is additionally C^2 differentiable, then accelerations $\ddot{r}_{\mathbb{S}}(t)$ are continuous as well. In general, if f is C^n differentiable, consequently $r^{(n)}_{\mathbb{S}}(t)$ values are continuous, unless directly perturbed.

Table 4.1 summarizes the needed formulae to create continuous-time morphed oscillators, and also gives a simple example. One only needs to define a base oscillator, and choose the desired limit cycle $\Xi_{\mathbb{S}}(\theta)$ to obtain the compact equations of the desired morphed oscillator.

Note: From now on and for the rest of this chapter: 1) We will use the compensated form (Equations (4.5,4.8)) to extend for higher dynamical system orders and dimension (our experience shows that the compensated form is less sensitive to the choice of numerical integration method). It is possible to rewrite the same procedures for the original form in the same way; 2) We remove the time indexing "$.(t)$" for brevity, and will only mention time if there is an explicit time dependency. So terms like $r_{\mathbb{S}}(t)$, $\theta(t)$ and $f(\theta(t))$ will respectively become $r_{\mathbb{S}}$, θ and $f(\theta)$; 3) All the phase portraits are illustrated only for $r_{\mathbb{S}} > 0$, unless mentioned otherwise.

4.2.2 Second Order Dynamical System

Equations (4.5, 4.8) define a first order dynamical system. Equation (4.5) drives the phase dynamics while Equation (4.8) controls the radial dynamics. If one desires to directly control acceleration, or create the possibility to add feedback mechanisms on the acceleration dynamics, then a second order dynamical system is needed. One can rewrite Equation (4.8) for a new velocity state ($v_{\mathbb{S}}$), and take the velocity profile, instead of position profile, as the limit cycle. The desired velocity limit cycle is $\dot{\Xi}_{\mathbb{S}}(\theta)$, so:

$$\begin{aligned} f_v(\theta) &= \frac{\dot{\Xi}_{\mathbb{S}}(\theta)}{\Xi_{\mathbb{B}}(\theta)} = \frac{\dot{\Xi}_{\mathbb{B}}(\theta)}{\Xi_{\mathbb{B}}(\theta)} f(\theta) + \dot{f}(\theta) \\ \dot{v_{\mathbb{S}}} &= \Xi_{\mathbb{B}}(\theta)\dot{f}_v(\theta) + \big(f_v(\theta)+\delta\big)\mathscr{D}_{\mathbb{B},r}\left(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}\right) \end{aligned} \tag{4.9}$$

where a constant $\delta > -\min_{\theta}\big(f_v(\theta)\big)$ is added to $f_v(\theta)$ to keep the scaling function strictly positive (to satisfy $\forall\theta : f(\theta) \neq 0$ needed for diffeomorphism). The oscillator obtained by Equations (4.5,4.9) will follow a desired velocity profile, but yet we need an equation for $\dot{r_{\mathbb{S}}}$. Additionally, unwanted position offsets (errors on $r_{\mathbb{S}} = \int v_{\mathbb{S}}\, dt$) are not forgotten in the system defined

Table 4.1: Continuous-time first order morphed oscillators. Two forms of oscillators are obtained: the original form, and the compensated form. $\Xi_{\mathbb{S}}(\theta)$ is the desired limit cycle, $\Xi_{\mathbb{B}}(\theta)$ is the limit cycle of the base oscillator, $f(\theta)$ is the phase-based scaling function, ω is the oscillation frequency multiplied by 2π, γ controls the rate of convergence, θ is the phase of the oscillator, and $r_{\mathbb{S}}$ is the radial state and also the desired output of the system which converges to the desired limit cycle $\Xi_{\mathbb{S}}(\theta)$.

	Original form	Compensated form
Morphed oscillator equations	$\dot{\theta} = \omega$ $\dot{r_{\mathbb{S}}} = \frac{r_{\mathbb{S}}}{f(\theta)}\dot{f}(\theta) + f(\theta)\mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}}{f(\theta)}\right)$	$\dot{\theta} = \omega$ $\dot{r_{\mathbb{S}}} = \Xi_{\mathbb{B}}(\theta)\dot{f}(\theta) + f(\theta)\mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}}{f(\theta)}\right)$
Example: desired limit cycle $\Xi_{\mathbb{S}}(\theta) = e^{sin(\theta)}$ and base oscillator $\mathscr{D}_{\mathbb{B},r}(r) = \gamma(\mu - r)$ (so $f(\theta) = \frac{\Xi_{\mathbb{S}}(\theta)}{\Xi_{\mathbb{B}}(\theta)} = \frac{1}{\mu}e^{sin(\theta)}$)	$\dot{\theta} = \omega$ $\dot{r_{\mathbb{S}}} = \omega\cos(\theta)r_{\mathbb{S}} + \gamma(e^{sin(\theta)} - r_{\mathbb{S}})$	$\dot{\theta} = \omega$ $\dot{r_{\mathbb{S}}} = \omega\cos(\theta)e^{sin(\theta)} + \gamma(e^{sin(\theta)} - r_{\mathbb{S}})$

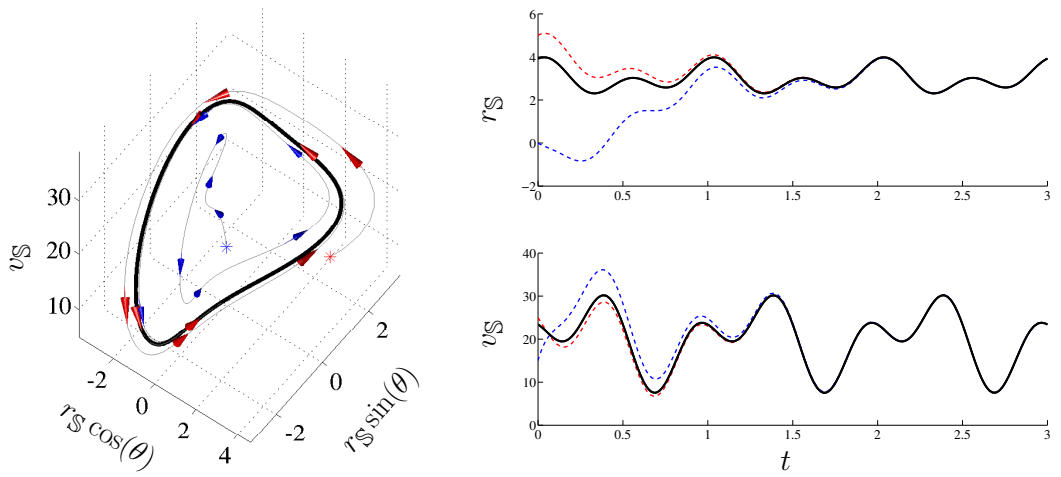


Figure 4.3: An example second order system. The desired limit cycle is $\Xi_{\mathbb{S}}(\theta) = 3 + 0.5(\cos(\theta) + \sin(2\theta+1))$. The base oscillator is defined by $\dot{r}_{\mathbb{B}} = 2\pi\vartheta\cos(\theta)+2+\sin(\theta)-r_{\mathbb{B}}$, which has the limit cycle $\Xi_{\mathbb{B}}(\theta) = 2+\sin(\theta)$. Consequently $f(\theta) = \frac{\Xi_{\mathbb{S}}(\theta)}{2+\sin(\theta)}$. For this figure $\gamma = 10$, $\vartheta = 1$ and $\beta = 20$. Asterix $(*)$ symbols indicate the initial points in the trajectories. Blue and red trajectories are the same for state-time and phase plots.

by Equation (4.9). We add a phase-based attractor force field to $\dot{v}_{\mathbb{S}}$ to damp the unwanted position offsets, and attract to the desired limit cycle in $\mathbb{S}$, which is $\Xi_{\mathbb{S}}(\theta) = \Xi_{\mathbb{B}}(\theta) f(\theta)$:

$$\dot{v}_{\mathbb{S}} = \Xi_{\mathbb{B}}(\theta)\dot{f}_v(\theta) + \big(f_v(\theta)+\delta\big)\mathscr{D}_{\mathbb{B},r}\left(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}\right) + \beta\left(\Xi_{\mathbb{B}}(\theta)f(\theta) - r_{\mathbb{S}}\right) \tag{4.10}$$

$$\dot{r}_{\mathbb{S}} = v_{\mathbb{S}} - \delta\Xi_{\mathbb{B}}(\theta) \tag{4.11}$$

where β determines the strength of the attractor force field defined by $\Xi_{\mathbb{B}}(\theta)f(\theta) - r_{\mathbb{S}}$ term. Also since a δ offset is added to f_v, the velocity limit cycle is having an offset of $\delta\Xi_{\mathbb{B}}(\theta)$, which is subtracted from $v_{\mathbb{S}}$ in Equation (4.11). Equations (4.5,4.10,4.11) together give a general second order phase oscillator which exhibits a desired limit cycle behavior. It should be noted that as long as the base oscillator's limit cycle is circular $\dot{\Xi}_{\mathbb{B}}(\theta) = 0$, and Equations (4.10,4.11) can be written as a simpler form where $f_v(\theta) = \dot{f}(\theta)$ and $\dot{f}_v(\theta) = \ddot{f}(\theta)$. Figure 4.3 illustrates an example state time evolution and phase portrait of a compensated second-order system.

4.2.3 $n-$th Order Dynamical System

The idea of extending to a second order system can be generalized to create an $n-$th order dynamical system. The $n-$th order radial state is controlled by the base oscillator. All radial

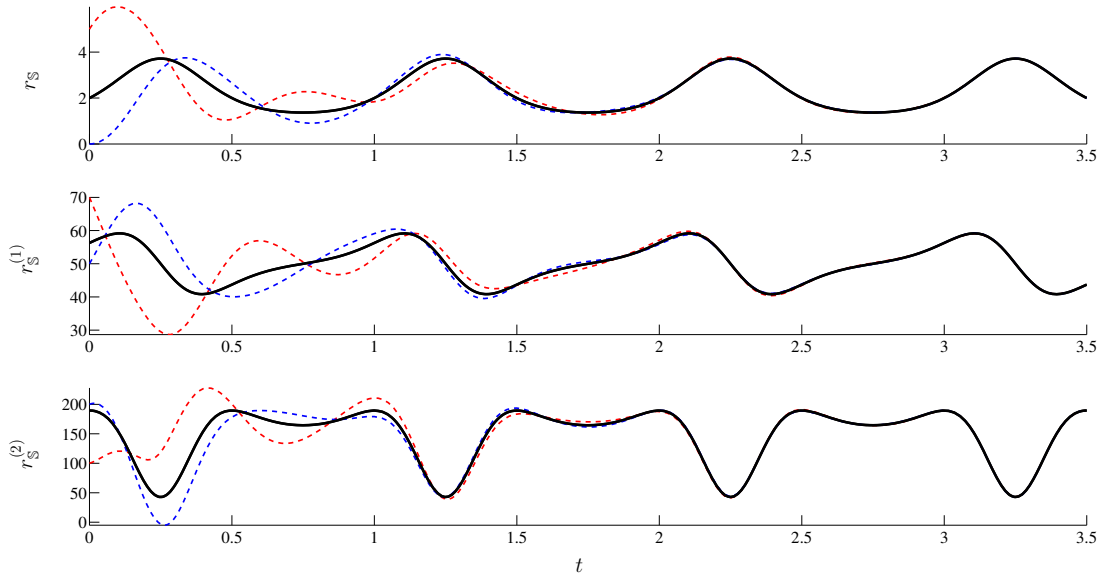


Figure 4.4: Example state-time evolution of a 3rd order system. The base oscillator is defined with $\dot{\theta} = 2\pi\vartheta$, $\dot{r}_{\mathbb{B}} = \gamma \tanh(\mu - r_{\mathbb{B}})$. The desired limit cycle is $\Xi_{\mathbb{S}}(\theta) = 1 + e^{\sin(\theta)}$. For this example $\vartheta = 1$, $\mu = 1$, $\gamma = 10$, $\beta_1 = \beta_2 = 50$, $\delta_1 = 50$ and $\delta_2 = 150$.

differential equations are augmented by attractors from states which are one order lower ($\dot{r}_{\mathbb{S}}^{m}$ is augmented with an attractor on $r_{\mathbb{S}}^{(m-1)}$):

$$
\begin{aligned}
\dot{r}_{\mathbb{S}}^{(n-1)} &= \Xi_{\mathbb{B}}(\theta)\dot{f}_{(n-1)}(\theta) \\
&+ \left(f_{(n-1)}(\theta) + \delta_{n-1}\right)\mathscr{D}_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}^{(n-1)}}{f_{(n-1)} + \delta_{n-1}(\theta)}\right) + \beta_{n-1}\left(\Xi_{\mathbb{B}}(\theta) f_{(n-2)}(\theta) + \delta_{n-2} - r_{\mathbb{S}}^{(n-2)}\right) \\
\dot{r}_{\mathbb{S}}^{(n-2)} &= r_{\mathbb{S}}^{(n-1)} - \delta_{n-1}\Xi_{\mathbb{B}}(\theta) + \beta_{n-2}\left(\Xi_{\mathbb{B}}(\theta) f_{(n-3)}(\theta) + \delta_{n-3} - r_{\mathbb{S}}^{(n-3)}\right) \\
\dot{r}_{\mathbb{S}}^{(n-3)} &= r_{\mathbb{S}}^{(n-2)} - \delta_{n-2} + \beta_{n-3}\left(\Xi_{\mathbb{B}}(\theta) f_{(n-4)}(\theta) + \delta_{n-4} - r_{\mathbb{S}}^{(n-4)}\right) \\
&\vdots \\
\dot{r}_{\mathbb{S}}^{(1)} &= r_{\mathbb{S}}^{(2)} - \delta_2 + \beta_1\left(\Xi_{\mathbb{B}}(\theta) f(\theta) - r_{\mathbb{S}}\right) \\
\dot{r}_{\mathbb{S}} &= r_{\mathbb{S}}^{(1)} - \delta_1
\end{aligned}
\tag{4.12}
$$

where $f_{(m)}(\theta) = \frac{d^m \Xi_{\mathbb{S}}(\theta)}{dt^m}\frac{1}{\Xi_{\mathbb{B}}(\theta)}$, $\delta_m > -\min_{\theta}\left(f_{(m)}(\theta)\right)$, and β_m is the strength of the force field on the m-th order which damps the unwanted offsets of $r_{\mathbb{S}}^{(m-2)}$. $r_{\mathbb{S}}, r_{\mathbb{S}}^{(1)}, \ldots, r_{\mathbb{S}}^{(n-1)}$ are system states corresponding respectively to position, velocity, acceleration, and so on. Figure 4.4 depicts an example state-time evolution for a 3rd order system.

Table 4.2: Example first order realizations with different base oscillators.

Base oscillator	Compensated first order realization
(I) Amplitude controlled oscillator $\dot{r}_{\mathbb{B}} = \gamma\left(\mu - r_{\mathbb{B}}\right)$	$\dot{r_{\mathbb{S}}} = \mu\dot{f}(\theta) + \gamma\left(\mu f(\theta) - r_{\mathbb{S}}\right)$
(II) Hopf oscillator $\dot{r}_{\mathbb{B}} = \gamma(\mu - r_{\mathbb{B}}^2) r_{\mathbb{B}}$	$\dot{r_{\mathbb{S}}} = \sqrt{\mu}\dot{f}(\theta) + \gamma\left(\mu - \left(\frac{r_{\mathbb{S}}}{f(\theta)}\right)^2\right) r_{\mathbb{S}}$
(III) Logarithmic saturated oscillator $\dot{r}_{\mathbb{B}} = \gamma \tanh(\mu - r_{\mathbb{B}}^2)\log(1 + r_{\mathbb{B}}^2)$	$\dot{r_{\mathbb{S}}} = \sqrt{\mu}\dot{f}(\theta) + \gamma\dot{f}(\theta)\tanh\left(\mu - \left(\frac{r_{\mathbb{S}}}{f(\theta)}\right)^2\right)\log\left(1 + \left(\frac{r_{\mathbb{S}}}{f(\theta)}\right)^2\right)$
(IV) Morphed oscillator $\dot{r}_{\mathbb{B}} = \omega\cos(\theta) + 2 + \sin(\theta) - r_{\mathbb{B}}$	$\dot{r_{\mathbb{S}}} = (2 + sin(\theta))\,\dot{f}(\theta) + f(\theta)\left(\omega\cos(\theta) + 2 + \sin(\theta) - \frac{r_{\mathbb{S}}}{f(\theta)}\right)$

Table 4.3: Example second order realizations with different base oscillators.

Base oscillator	Compensated second order realization
(I) Amplitude controlled oscillator $\dot{r}_{\mathbb{B}} = \gamma\left(\mu - r_{\mathbb{B}}\right)$	$\dot{v_{\mathbb{S}}} = \mu\ddot{f}(\theta) + \gamma\left(\mu\dot{f}(\theta) + \mu\delta - v_{\mathbb{S}}\right) + \beta(\mu f(\theta) - r_{\mathbb{S}})$ $\dot{r}_{\mathbb{S}} = v_{\mathbb{S}} - \mu\delta$
(II) Hopf oscillator $\dot{r}_{\mathbb{B}} = \gamma(\mu - r_{\mathbb{B}}^2) r_{\mathbb{B}}$	$\dot{v_{\mathbb{S}}} = \sqrt{\mu}\ddot{f}(\theta) + \gamma\left(\mu - \left(\frac{v_{\mathbb{S}}}{\dot{f}(\theta)+\delta}\right)^2\right) v_{\mathbb{S}} + \beta(\sqrt{\mu} f(\theta) - r_{\mathbb{S}})$ $\dot{r}_{\mathbb{S}} = v_{\mathbb{S}} - \sqrt{\mu}\delta$
(III) Logarithmic saturated oscillator $\dot{r}_{\mathbb{B}} = \gamma \tanh(\mu - r_{\mathbb{B}}^2)\log(1 + r_{\mathbb{B}}^2)$	$\dot{v}_{\mathbb{S}} = \sqrt{\mu}\ddot{f}(\theta)$ $+\gamma(\dot{f}(\theta) + \delta)\tanh\left(\mu - \left(\frac{v_{\mathbb{S}}}{\dot{f}(\theta)+\delta}\right)^2\right)\log\left(1 + \left(\frac{v_{\mathbb{S}}}{\dot{f}(\theta)+\delta}\right)^2\right) + \beta(\sqrt{\mu} f(\theta) - r_{\mathbb{S}})$ $\dot{r}_{\mathbb{S}} = v_{\mathbb{S}} - \sqrt{\mu}\delta$
(IV) Morphed oscillator $\dot{r}_{\mathbb{B}} = \omega\cos(\theta) + 2 + \sin(\theta) - r_{\mathbb{B}}$	$f_v(\theta) = \frac{\omega\cos(\theta)}{2+\sin(\theta)} f(\theta) + \dot{f}(\theta)$ $\dot{v}_{\mathbb{S}} = (2 + sin(\theta))\,\dot{f}_v(\theta)$ $+\gamma(f_v(\theta) + \delta)\left(\omega\cos(\theta) + 2 + \sin(\theta) - \frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}\right) + \beta\left(\left(2 + \sin(\theta)\right) f(\theta)\right) - r_{\mathbb{S}}\right)$ $\dot{r}_{\mathbb{S}} = v_{\mathbb{S}} - (2 + \sin(\theta))\delta$

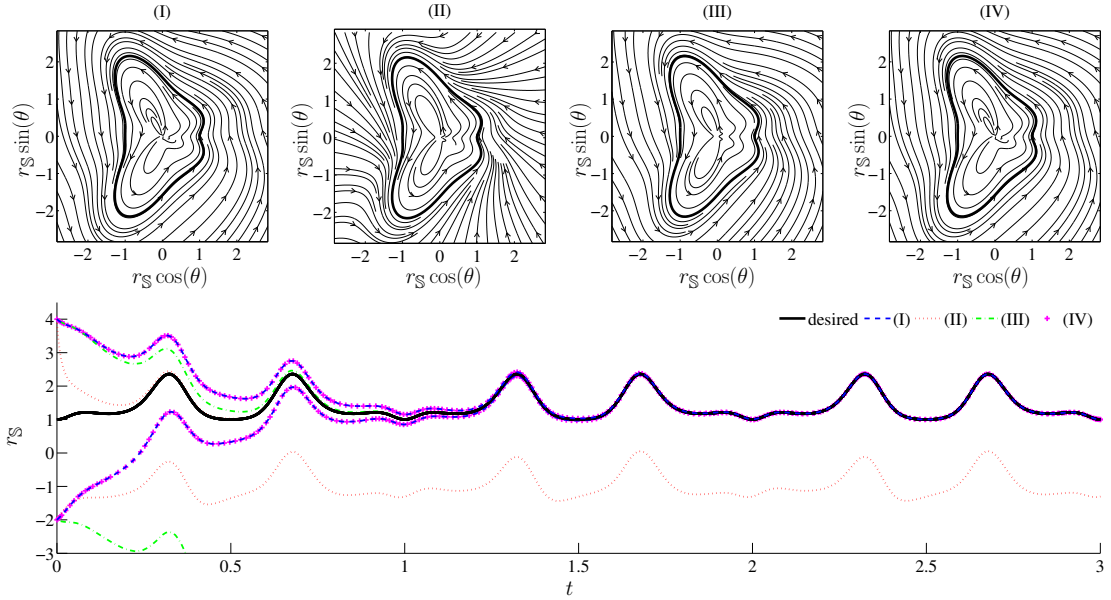


Figure 4.5: First order realization examples with four different base oscillators: (I) amplitude controlled oscillator, (II) Hopf oscillator, (III) logarithmic saturated oscillator, (IV) morphed oscillator. The shape of limit cycle is the same for all examples, and the difference is in the convergence behavior. The initial state value in the bottom plot, $r_{\mathbb{S}} = 4$, is corresponding to $r_{\mathbb{S}}\cos(\theta) = 4, r_{\mathbb{S}}\sin(\theta) = 0$ in the top plots ($\theta|_{t=0} = 0$). The basin of attraction of the desired limit cycle for (I) and (IV) is the whole state space, while this is not true for (II) and (III). The basin of attraction for these oscillators are later defined in Table 4.4. The behavior of examples (I) and (IV) are the same. This is because example (IV) uses a morphed oscillator as base which has been obtained by morphing an amplitude controlled oscillator, the base of (I). So (IV) is identical to having an amplitude controlled oscillator as base, and then morphing it twice using two phase-based scaling functions.

4.3 Realization

The previous section gave a general methodology to design morphed phase oscillators with arbitrary limit cycle shapes. The obtained forms represent a general family of morphed oscillators that are parametrized by the radial equation of the base oscillator $\mathscr{D}_{\mathbb{B},r}(.)$. Realizations are obtained by using desired base oscillators, defining $\mathscr{D}_{\mathbb{B},r}(.)$.

Tables 4.2-4.3 show first and second order realizations of this methodology with different base oscillators. First a base oscillator is chosen, and then Equation (4.8) or Equations (4.10,4.11) are used to obtain the morphed oscillator. For all the examples given $\dot{\theta} = 2\pi\vartheta$, γ controls the rate of convergence, and μ corresponds to the radius of the limit cycle in $\mathbb{B}$.

Figure 4.5 illustrates phase portraits of these oscillators as well as their state-time evolution (both for first order realizations). The desired limit cycle $\Xi_{\mathbb{S}}(\theta)$ is chosen to be the same for all the examples to enable comparison. One should keep in mind that although $\Xi_{\mathbb{S}}(\theta)$ is the same for all the examples, the corresponding f functions are not necessarily the same, because

$\Xi_{\mathbb{B}}(\theta)$ is different for different base oscillators (and we know $f(\theta) = \frac{\Xi_{\mathbb{S}}(\theta)}{\Xi_{\mathbb{B}}(\theta)}$).

The basin of attraction in examples (I) and (IV) is the whole state space, while this is not true for examples (II) and (III). Example (II) uses a Hopf oscillator as base, which has two limit cycles at $r_{\mathbb{B}}(\theta) = \pm\sqrt{\mu}$, and states who enter the basin of attraction of the unwanted limit cycle at $r_{\mathbb{B}}(\theta) = -\sqrt{\mu}$ will not converge to the desired limit cycle. Finally, for example (III), the solution diverges initiating with $r_{\mathbb{S}}|_{t=0} = -2$. We will discuss the stability conditions in Section 4.5 and see when the oscillator in example (III) gets unstable.

4.3.1 Equivalence

Here we will show how certain realizations of the morphed oscillators represent familiar trajectory control/generation methods. The first example in Table 4.3 implements a second order dynamical system which represents a form of the rhythmic Dynamical Movement Primitives [Ijspeert et al., 2013]. Dynamical Movement Primitives (DMP) are robust movement generators that are commonly used for motor control. The rhythmic DMP is formulated as:

$$
\begin{aligned}
\tau\dot{z} &= \alpha_z(\beta_z(g-y)-z)+F \\
\tau\dot{y} &= z && (4.13)\\
\tau\dot{\theta} &= 1 && (4.14)\\
F &= \tau^2\ddot{y}_{des}(t)+\tau\alpha_z\dot{y}_{des}(t)+\alpha_z\beta_z y_{des}(t)-\alpha_z\beta_z g \\
&= \tau^2(\frac{1}{\tau})^2 y''_{des}(\theta)+\tau(\frac{1}{\tau})\alpha_z y'_{des}(\theta)+\alpha_z\beta_z y_{des}(\theta)-\alpha_z\beta_z g && (4.15)
\end{aligned}
$$

where τ is the cycle period divided by 2π, and F is the nonlinear forcing term to shape the limit cycle such that the output y_{des} is obtained. Rewriting the transformation system (Equation (4.13)) with expanded F and simplifying gives:

$$
\begin{aligned}
\dot{z} &= \tau\ddot{y}_{des}(\theta)+\frac{\alpha_z}{\tau}(\tau\dot{y}_{des}(\theta)-z)+\frac{\alpha_z\beta_z}{\tau}(y_{des}(\theta)-y) \\
\dot{y} &= \frac{1}{\tau}z && (4.16)
\end{aligned}
$$

and by assuming $r_{\mathbb{S}} = y$, $v_{\mathbb{S}} = \frac{1}{\tau}z$ and $f(\theta) = y_{demo}(\theta)$, the above is identical to a second-order realization with a unit radius amplitude controlled oscillator as base ($\mu = 1$), $\gamma = \frac{\alpha_z}{\tau}$ and $\beta = \frac{\alpha_z\beta_z}{\tau^2}$. So with the right representation, rhythmic DMPs (with the latest representation in [Ijspeert et al., 2013]) are a subset of the general family generated by the compensated form.

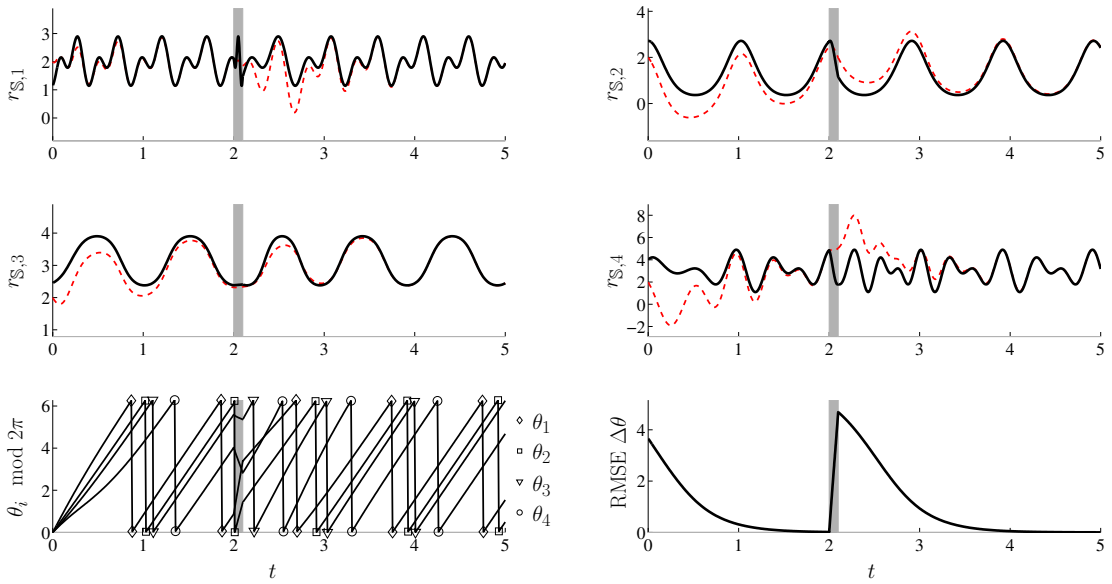


Figure 4.6: A coupled four dimensional system of second order morphed oscillators. The desired phase differences, with respect to θ_1, are $\{0, \pi/3, \pi/2, \pi\}$. The base oscillators are respectively the ones in Table 4.3. The desired limit cycles are $\Xi_{\mathbb{S},1}(\theta) = 2 + \cos(3\theta) + sin(\theta - 1)$, $\Xi_{\mathbb{S},2}(\theta) = e^{\cos(\theta)}$, $\Xi_{\mathbb{S},3}(\theta) = \pi + \tanh(\sin(\theta - 1))$, and $\Xi_{\mathbb{S},4}(\theta) = \sin(2\theta) + cos(3\theta) + 3$. The top four plots depict the evolution of the output state $r_{\mathbb{S},i}$ over time, for each dimension. The black references are $\Xi_{\mathbb{S},i}(\theta)$ and the dashed red trajectories are the outputs of each dimension $r_{\mathbb{S},i}$. The bottom-left plot shows the evolution of the phases of the oscillators. All phases are initialized as 0, and they get coupled around $t = 1.5$. The phases are perturbed for $t \in [2, 2.1]$. The bottom-right figure shows the root mean square error between the vector of phases and the vector of desired phase differences. One can see that phases get coupled again after the perturbation at about $t = 4$. For this example all $\gamma_i = 10$, $\beta_i = 20$, $\delta_i = 10$, $\vartheta = 1$, and $c_{ik} = 1$.

4.4 Extension to Higher Dimensions

The methodology presented in Section 4.2 gives a systematic way to create nonlinear oscillators with one dimensional outputs $r_{\mathbb{S}}$. In this section we explain how multidimensional oscillators can be created out of these one dimensional oscillators. Extending to high dimensions expands the scope which the introduced oscillators can be applied, including coupled synchronized high dimensional movements needed in applications like robotics [Dégallier et al., 2006, Fukuoka et al., 2003], or in modeling of Central Pattern Generators [Matsuoka, 1987, Ijspeert, 2008] .

Several approaches including [Ajallooeian et al., 2012, Ijspeert et al., 2007, Righetti and Ijspeert, 2006b] use coupled oscillators to create multidimensional nonlinear oscillators. One can use diffusive phase coupling [Strogatz, 2000] to create the multidimensional system, so Equation

(4.5) changes to:

$$\dot{\theta}_i = \omega + \sum_{k=1}^{N} c_{ik} \sin(\theta_k - \theta_i - \phi_{ik}) \tag{4.17}$$

where θ_i is the phase state of the $i-$th oscillator, c_{ik} is the coupling strength between $i-$th and $k-$th oscillators, and ϕ_{ik} is the desired phase difference between them. The diffusive schema in Equation (4.17) couples the phase dynamics, and this coupling is independent of the output states $r_{\mathbb{S},i}$. Implementing coupling schema which are affected by the $r_{\mathbb{S},i}$ states is possible, and is application specific. An example can be found in [Ijspeert et al., 2007].

Figure 4.6 illustrates an all-to-all coupled four dimensional system where each oscillator is a second-order morphed oscillator. A different limit cycle shape is chosen for each dimension, and these are depicted in the top four plots. The bottom-left plot shows the phase-time evolution, and phase differences can be seen with the overlaid markers. All phases are initialized with 0 value, and they are also perturbed for $t \in [2, 2.1]$. The system gradually gets synchronized after the perturbation, as illustrated in the bottom-right plot in Figure 4.6.

4.5 Stability

This section is dedicated to stability analysis of the introduced morphed oscillators. We first discuss the stability of a one dimensional first order system using the Poincaré-Bendixson theorem [Guckenheimer and Holmes, 1983]. Analysis of $n-$th order systems, $n > 1$, is complex as the Poincaré-Bendixson theorem is not valid anymore, and we briefly discuss these systems using Contraction Theory [Lohmiller and Slotine, 1998]. Finally we include the stability of the multidimensional system.

Before going into the stability analysis, we would like to describe the space formed by $\{\theta, r_{\mathbb{S}}\}$. One can simply assume that $\theta \in [0, 2\pi)$, $r_{\mathbb{S}} \in \mathbb{R}^+$, and by doing so $\{\theta, r_{\mathbb{S}}\}$ forms the standard polar coordinates. However we additionally want to be able to analyze the system when $r_{\mathbb{S}} < 0$ (without jumping to the antiphase state $\{\theta + \pi, |r_{\mathbb{S}}|\}$). One can assume that $\theta \in [0, 2\pi)$, $r_{\mathbb{S}} \in \mathbb{R}^+$ forms a manifold, and $\theta \in [0, 2\pi)$, $r_{\mathbb{S}} \in \mathbb{R}^-$ forms a second manifold, and these two manifolds are connected when $r_{\mathbb{S}} = 0$. The resulting manifold can be chosen to be a 2-manifold, as illustrated in Figure 4.7, to describe the space formed by $\{\theta, r_{\mathbb{S}}\} \in [0, 2\pi) \times \mathbb{R}$. We call this representation the *extended polar coordinates* where negative radius values are meaningful. The chosen 2-manifold in Figure 4.7 is only an arbitrary representation, and any other representation which defines an orientable 2-manifold is valid.

4.5.1 One Dimensional First Order System

Analyzing the limit cycle properties of a dynamical system is not an easy task. The analysis becomes possible when one is looking for the existence of limit cycles in a bounded region of a phase plane, where the Poincaré-Bendixson's theorem can be utilized. The same theorem

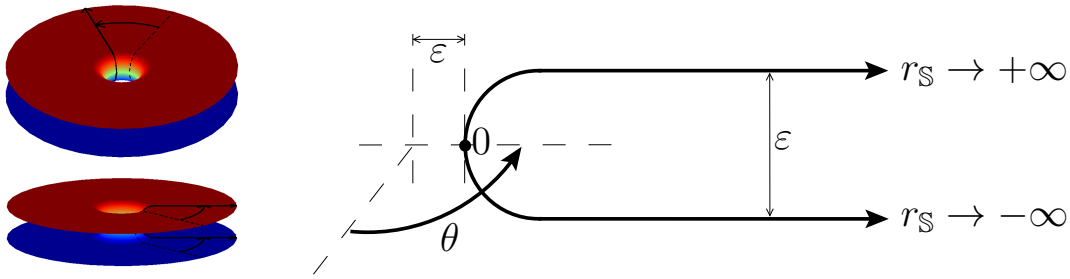


Figure 4.7: A representative 2-manifold formed by $\theta \in [0, 2\pi)$, $r_{\mathbb{S}} \in \mathbb{R}$. When the value of $r_{\mathbb{S}}$ crosses zero the system state goes from one disk to another. ε is a positive constant. The pulley-like 2-manifolds on the left are obtained by rotational sweep of the coordinates shown in right.

can also be used the analyze the asymptotic stability properties of a limit cycle system. The original form of this theorem is:

Theorem (Poincaré-Bendixson [Guckenheimer and Holmes, 1983])**.** *Every nonempty, compact ω-limit set of a C^1 planar flow that does not contain an equilibrium point is a (nondegenerate) periodic orbit.*

A simpler interpretation of this theorem is: given a differential equation in the plane, assume $\zeta(t)$ is a solution curve which stays in a bounded region. Then, if there is no equilibrium point in this region, $\zeta(t)$ converges for $t \to +\infty$ to a periodic trajectory. Now if there is only one periodic orbit in this region, the asymptotic stability of the corresponding limit cycle in this bounded region is assured. Poincaré-Bendixson's theorem also holds for every orientable 2-manifold for which the Jordan curve theorem [Fulton, 1995] holds. The extended polar coordinates defined by $\{\theta, r_{\mathbb{S}}\}$ is an orthogonal coordinate system, and the manifold created by $\{\theta, r_{\mathbb{S}}\} \in \mathbb{R}^2$ is an orientable 2-manifold satisfying Jordan curve theorem. To employ Poincaré-Bendixson's theorem on our problem, we need to define a bounded region around the desired limit cycle $\Xi_{\mathbb{S}}(\theta)$. We define the **L**ower and **U**pper bounds around $\Xi_{\mathbb{S}}(\theta)$ as:

$$\begin{aligned} g_L &: \theta \longmapsto g_L(\theta)\ ;\ g_L(\theta) < \Xi_{\mathbb{S}}(\theta) \\ g_U &: \theta \longmapsto g_U(\theta)\ ;\ g_U(\theta) > \Xi_{\mathbb{S}}(\theta) \end{aligned} \tag{4.18}$$

For the upper bound, vectors pointing inwards the bounded region are defined as $\mathbf{p} = \{\dot{g}_U(\theta), -\dot{\theta}\}$, and for the lower bound they are $\mathbf{p} = \{-\dot{g}_L(\theta), \dot{\theta}\}$ (assuming clockwise phase

evolution, i.e. $\dot{\theta} > 0$). The system dynamics on these bounds are also defined as:

$$\begin{aligned} \text{org. form, Equation (4.7)} \quad &: \quad \mathbf{d} = \{\dot{\theta}, \frac{g_\iota(\theta)}{f(\theta)}\dot{f}(\theta) + f(\theta)\mathscr{D}_{\mathbb{B},r}(\frac{g_\iota(\theta)}{f(\theta)})\}, \iota \in \{L,U\} \\ \text{com. form, Equation (4.8)} \quad &: \quad \mathbf{d} = \{\dot{\theta}, \Xi_{\mathbb{B}}(\theta)\dot{f}(\theta) + f(\theta)\mathscr{D}_{\mathbb{B},r}(\frac{g_\iota(\theta)}{f(\theta)})\}, \iota \in \{L,U\} \end{aligned} \tag{4.19}$$

To have the condition that no flow leaves the bounds we need (with $\langle .,. \rangle$ operator being the inner product):

$$\langle \mathbf{p}, \mathbf{d} \rangle > 0 \tag{4.20}$$

To utilize the Poincaré-Bendixson's theorem, it is needed to define bounds such that no flows leaves the enclosed area. Figure 4.8 gives an idea about how to define the bounds. For the original form we define:

$$g_{L|U}(\theta) = \Xi_{\mathbb{S}}(\theta) + \kappa f(\theta) = (\Xi_{\mathbb{B}}(\theta) + \kappa) f(\theta); \quad \kappa \lessgtr 0 \tag{4.21}$$

Rewriting Equation (4.20) and dividing by the positive term $\dot{\theta} f(\theta)$ gives (by definition $f(\theta) > 0$ and we have already assumed $\dot{\theta} > 0$):

$$\text{lower bound} \quad : \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \kappa) > \dot{\Xi}_{\mathbb{B}}(\theta); \quad \kappa < 0 \tag{4.22}$$

$$\text{upper bound} \quad : \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \kappa) < \dot{\Xi}_{\mathbb{B}}(\theta); \quad \kappa > 0 \tag{4.23}$$

Now if the conditions in Equations (4.22-4.23) are met for all possible margins in a bounded region, then the desired limit cycle is asymptotically stable. So the basin of attraction for the original form is defined by the bounds:

$$\begin{aligned} \kappa_L \in \mathbb{R} \mid \forall\theta \in \mathbb{R} \;\&\; \forall\kappa,\, \kappa_L < \kappa < 0 \quad &: \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \kappa) > \dot{\Xi}_{\mathbb{B}}(\theta) \\ \kappa_U \in \mathbb{R} \mid \forall\theta \in \mathbb{R} \;\&\; \forall\kappa,\, 0 < \kappa < \kappa_U \quad &: \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \kappa) < \dot{\Xi}_{\mathbb{B}}(\theta) \end{aligned} \tag{4.24}$$

The lower and upper bounds for the compensated form in Equation (4.8) are not same as the ones of the original form. This is depicted in Figure 4.8. What happens with the compensated form is that the dynamics contours are like the shape of limit cycle for states near it, but become circular for states far from limit cycle. So we define:

$$g_{L|U}(\theta) = \Xi_{\mathbb{S}}(\theta) + \kappa = \Xi_{\mathbb{B}}(\theta) f(\theta) + \kappa; \quad \kappa \lessgtr 0 \tag{4.25}$$

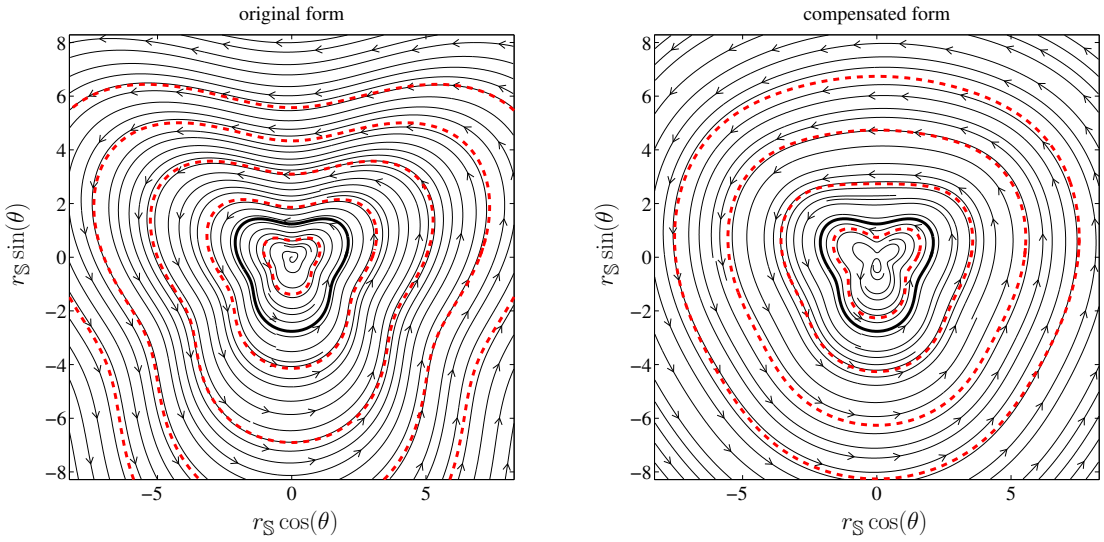


Figure 4.8: Poincaré-Bendixson bounds (dashed red) for the first order original and compensated forms. The bounds of the original form are scaled versions of the limit cycle, while the bounds of the compensated form are similar to the limit cycle shape when near it, and become circular when far. For this example $\Xi_{\mathbb{S}}(\theta) = 2 + \sin(\theta)\tanh(\cos(2\theta))$, and the base is an amplitude controlled oscillator $\vartheta = 1$, $\mu = 1$ and $\gamma = 0.2$. The trajectories of the system dynamics (solid gray) enter the bounds (dashed red) and do not leave them.

and by using the same procedure used for the original form we get:

$$\mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \frac{\kappa}{f(\theta)}) \gtrless \dot{\Xi}_{\mathbb{B}}(\theta); \quad \kappa \lesseqgtr 0 \tag{4.26}$$

The term $\frac{\kappa}{f(\theta)}$ is strictly positive when $\kappa > 0$ and strictly negative if $\kappa < 0$. So again the basin of attraction is defined by the bounds:

$$\begin{aligned} \kappa_L \in \mathbb{R} \mid \forall\theta \in \mathbb{R} \;\&\; \forall\kappa,\, \kappa_L < \kappa < 0 \quad &: \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \frac{\kappa}{f(\theta)}) > \dot{\Xi}_{\mathbb{B}}(\theta) \\ \kappa_U \in \mathbb{R} \mid \forall\theta \in \mathbb{R} \;\&\; \forall\kappa,\, 0 < \kappa < \kappa_U \quad &: \quad \mathscr{D}_{\mathbb{B},r}(\Xi_{\mathbb{B}}(\theta) + \frac{\kappa}{f(\theta)}) < \dot{\Xi}_{\mathbb{B}}(\theta) \end{aligned} \tag{4.27}$$

Table 4.4 gives examples of the basin of attraction for different realizations. The first four examples are the same morphed oscillators as in Table 4.2. The fifth example is with a base oscillator that has one stable limit cycle at $r_{\mathbb{B}} = \mu$ and two unstable ones at $r_{\mathbb{B}} = 0$ and $r_{\mathbb{B}} = 2\mu$. For the first four examples, the upper bound of the basin of attraction tends to infinity. The basin of attraction for the fifth example is bounded by the two other unstable limit cycles. In general, if the base has multiple limit cycles, the designer should choose which limit cycle is used for morphing, and the other limit cycles will then limit the span of the basin of attraction.

Table 4.4: The basin of attraction of first order realizations using different bases.

base oscillator	basin of attraction bounds parameters	
	original	compensated
$\dot{\theta} = \omega$ $\dot{r}_{\mathbb{B}} = \gamma(\mu - r_{\mathbb{B}})$	LB: $\kappa \to -\infty$ UB: $\kappa \to +\infty$	
$\dot{\theta} = \omega$ $\dot{r}_{\mathbb{B}} = \gamma(\mu - r_{\mathbb{B}}^2) r_{\mathbb{B}}$	LB: $\kappa = -\sqrt{\mu}$ UB: $\kappa \to +\infty$	LB: $\kappa = -\sqrt{\mu} \min_{\theta}\big(f(\theta)\big)$
$\dot{\theta} = \omega$ $\dot{r}_{\mathbb{B}} = \gamma \tanh(\mu - r_{\mathbb{B}}^2) \log(1 + r_{\mathbb{B}}^2)$	LB: $\kappa = -2\sqrt{\mu}$ UB: $\kappa \to +\infty$	LB: $\kappa = -2\sqrt{\mu} \min_{\theta}\big(f(\theta)\big)$
$\dot{\theta} = \omega$ $\dot{r}_{\mathbb{B}} = \omega \cos(\theta) + 2 + \sin(\theta) - r_{\mathbb{B}}$	LB: $\kappa \to -\infty$ UB: $\kappa \to +\infty$	
$\dot{\theta} = \omega$ $\dot{r}_{\mathbb{B}} = -\gamma\ r_{\mathbb{B}}(\mu - r_{\mathbb{B}})(2\mu - r_{\mathbb{B}})$	LB: $\kappa = -\mu$ UB: $\kappa = +\mu$	LB: $\kappa = -\mu \min_{\theta}\big(f(\theta)\big)$ UB: $\kappa = \mu \min_{\theta}\big(f(\theta)\big)$

4.5.2 One Dimensional Second Order System

Analyzing the asymptotic stability of a general nonlinear second order system is not trivial, and the Poincaré-Bendixson theorem cannot be used anymore as it is only valid for phase planes, and not volumes. We utilize Contraction Theory [Lohmiller and Slotine, 1998] to show the stability conditions of our 2nd+ order systems. Contraction Theory characterizes the system stability by the behavior of the differences between solutions with different initial conditions. If these differences vanish exponentially over time, all solutions converge towards a single trajectory, independent from the initial states. In this case, the system is called globally asymptotically stable. For a general dynamical system of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ assume that $\mathbf{x}(t)$ is one solution of the system, and $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) + \delta\mathbf{x}(t)$ a neighboring one with a different initial condition ($\delta\mathbf{x}(t)$ is also called virtual displacement). It can be shown that any nonzero virtual displacement decays exponentially to zero over time if the symmetric part of the Jacobian of $\mathbf{f}$ is uniformly negative definite. In this case, it can be shown that the norm of the virtual displacement decays at least exponentially to zero, for $t \to \infty$ [Lohmiller and Slotine, 1998]. Namely, $\|\delta\mathbf{x}\| \leq \exp \int_0^t \lambda_{max}(\mathbf{x}, t)\, dt \|\delta\mathbf{x}_0\|$, where $\lambda_{max}(\mathbf{x}, t)$ is the largest eigenvalue of $\frac{1}{2}(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^T)$, so that the sufficient contraction condition is: $\lambda_{max}(\mathbf{x}, t) \leq -b < 0$ (for some $b > 0$) uniformly in a contracting region.

One can write the infinitesimal virtual displacements of the second order compensated form as (since θ is indifferent and $f(\theta)$ is bounded, the phase-dependent terms can be considered as inputs over time):

$$\frac{d}{dt}\begin{bmatrix}\delta v_{\mathbb{S}} \\ \delta r_{\mathbb{S}}\end{bmatrix} = \mathbf{J}\begin{bmatrix}\delta v_{\mathbb{S}} \\ \delta r_{\mathbb{S}}\end{bmatrix} = \begin{bmatrix}\mathscr{D}'_{\mathbb{B},r}(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}) & -\beta \\ 1 & 0\end{bmatrix}\begin{bmatrix}\delta v_{\mathbb{S}} \\ \delta r_{\mathbb{S}}\end{bmatrix} \tag{4.28}$$

where $\mathbf{J}$ is the unsymmetrized Jacobian of Equations (4.10,4.11). This has a symmetrized

Jacobian:

$$\mathbf{J}_{sym} = \begin{bmatrix} \mathscr{D}'_{\mathbb{B},r}(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}) & (1-\beta)/2 \\ (1-\beta)/2 & 0 \end{bmatrix} \tag{4.29}$$

Since here the symmetrized Jacobian has its eigenvalues of the opposite signs, we apply the linear local coordinate transform for variational displacements, obtaining the new coordinates: $\delta v_{\mathbb{S}}$ and $\delta z_{\mathbb{S}} = \sqrt{\beta}\,\delta r_{\mathbb{S}}$ (for $\beta > 0$). Now the symmetrized Jacobian is

$$\mathbf{J}_{sym} = \begin{bmatrix} \mathscr{D}'_{\mathbb{B},r}(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta}) & 0 \\ 0 & 0 \end{bmatrix} \tag{4.30}$$

which has a zero eigenvalue and the other one is $\mathscr{D}'_{\mathbb{B},r}(\frac{v_{\mathbb{S}}}{f_v(\theta)+\delta})$. So the second order system is contracting when $\mathscr{D}'_{\mathbb{B},r}(.) < 0$, $\beta > 0$. With the above conditions, the compensated second order form is partially contracting towards the desired limit cycle and the phase subsystem is indifferent.

4.5.3 One Dimensional $n-$th Order System

The stability analysis of the $n-$th order system ($n > 2$) is similar to the one for a 2nd order system. Again writing the infinitesimal virtual displacements give:

$$\frac{d}{dt}\begin{bmatrix} \delta r_{\mathbb{S}}^{(n-1)} \\ \delta r_{\mathbb{S}}^{(n-2)} \\ \vdots \\ \delta r_{\mathbb{S}}^{(1)} \\ \delta r_{\mathbb{S}} \end{bmatrix} = \begin{bmatrix} D'_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}^{(n-1)}}{f_{(n-1)}+\delta_{n-1}}\right) & -\beta_{n-1} & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & -\beta_{n-2} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 & -\beta_1 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta r_{\mathbb{S}}^{(n-1)} \\ \delta r_{\mathbb{S}}^{(n-2)} \\ \vdots \\ \delta r_{\mathbb{S}}^{(1)} \\ \delta r_{\mathbb{S}} \end{bmatrix} \tag{4.31}$$

and again using the new coordinates: $\delta r_{\mathbb{S}}^{(n-1)}$, $\delta z_{\mathbb{S}}^{(n-2)} = \sqrt{\beta_{n-1}}\,\delta r_{\mathbb{S}}^{(n-2)}$, $\delta z_{\mathbb{S}}^{(n-3)} = \sqrt{\beta_{n-2}}\,\delta r_{\mathbb{S}}^{(n-3)}$, $\dots$, $\delta z_{\mathbb{S}} = \sqrt{\beta_1}\,\delta r_{\mathbb{S}}$, the $n-$th order system is partially contracting when $D'_{\mathbb{B},r}\left(\frac{r_{\mathbb{S}}^{(n-1)}}{f_{(n-1)}+\delta_{n-1}}\right) < 0$ and $\forall i : \beta_i > 0$.

4.5.4 Multidimensional System

The stability of the multidimensional system created in Equation (4.17) should be analyzed from two aspects: (I) asymptotic stability of the phase coupling, (II) asymptotic stability of each dimension under the coupling. Since phases are independent of the radial values, the multidimensional system is a hierarchy, and the asymptotic stability of phase coupling can be analyzed independently [Lohmiller and Slotine, 1998]. To address (I), it can be shown that as

long as for every loop in the coupling graph passing through oscillators $i_1, i_2, i_3, \ldots, i_m, i_1$:

$$\phi_{i_1 i_2} + \phi_{i_2 i_3} + \ldots + \phi_{i_m i_1} = 2k\pi, k \in \mathbb{Z} \tag{4.32}$$

with the additional condition $\forall i, j : c_{ij} \geq 0, c_{ij} = c_{ji}$, the phase differences $\theta_i - \theta_j$ will asymptotically converge to the desired phase differences ϕ_{ij}. One can introduce the potential function (for the coupling dynamics in Equation (4.17), with the change of variables $\theta_i \leftarrow \theta_i - \omega t$):

$$U(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}\cos(\theta_j - \theta_i - \phi_{ij}) \tag{4.33}$$

which gives: $\dot{\theta}_i = -\frac{\partial U}{\partial \theta_i}$ and the potential $U(\boldsymbol{\theta})$ has minima at $\forall i, j : \theta_i = \theta_j - \phi_{ij} + 2k\pi, \forall k \in \mathbb{Z}$. Since $\frac{dU}{dt} = \sum_{j=1}^{N} \frac{\partial U}{\partial \theta_j}\frac{d\theta_j}{dt} = -\sum_{j=1}^{N}(\frac{\partial U}{\partial \theta_j})^2$, then $U(\boldsymbol{\theta})$ plays a role of Lyapunov's function, proving the asymptotic stability. Now since the phase differences are consistent (Equation (4.32)), the system remains synchronized.

The multidimensional system is a hierarchically coupled system, where the radial dynamics depends on the phase dynamics but the phase dynamic does not depend on the radial dynamics, and the phase dynamics is contracting by itself. As long as the phases are synchronized, $\forall i : \dot{\theta}_i = \omega = 2\pi\vartheta$, then all oscillators will converge to their limit cycles. An asynchrony, e.g. $\theta_j - \theta_i \neq 2k\pi + \phi_{ij}$, can introduce a perturbation on the radial dynamics for the $i-$th dimension. Let us assume that this perturbation, at phase θ_i, is quantified as $u_i(\theta_i)$. As long as this perturbation does not push the $i-th$ oscillator out of its basin of attraction, it will eventually be forgotten, and the $i-$th oscillator will converge to its limit cycle. So in the case where the basin of attraction is the whole state space, the whole multidimensional system is asymptotically stable. For the case where the basin of attraction is limited, if the sum of the coupling weights is small enough, $u_i(\theta_i)$ will be small enough, and the system will remain in the basin of attraction. One can simply choose to have all the coupling weights equal and small enough to ensure stability.

4.6 Arbitrary Convergence Behavior

The previous sections introduced a methodology to design morphed phase oscillators with arbitrary limit cycle shapes, and analyzed their stability. The convergence behavior of these morphed oscillators is determined by the choice of the base oscillator, and cannot be explicitly defined. However it is useful to have an oscillator which can exhibit a given desired convergence behavior. Examples can be when a certain path should be followed to reach the limit cycle [Ernesti et al., 2012], or if phase-dependent convergence behavior is of interest.

To obtain such oscillator, we modify the case where a unit radius amplitude controlled oscillator is used as the base oscillator ($\Xi_{\mathbb{B}}(\theta) = 1$) and an original first order realization is applied. We first need to find the analytical solution for the case where an amplitude controlled oscillator

is used as the base. For the phase equation we can simply write:

$$\theta(t) = \omega t + C_\theta,\ C_\theta = \theta(0) \tag{4.34}$$

To find the analytical solution for the radius equation, we define $g = f(\theta)$, and we rewrite the morphed amplitude controlled oscillator (first example in Table 4.2) as (for simplicity, and without loss of generality, we assume $\mu = 1$):

$$\dot{r}_{\mathbb{S}} + \left(\gamma - \frac{\dot{g}}{g}\right) r_{\mathbb{S}} = \gamma g \tag{4.35}$$

This is a first order differential equation of the form:

$$\dot{r}_{\mathbb{S}} + p r_{\mathbb{S}} = q \tag{4.36}$$

with $p = \gamma - \frac{\dot{g}}{g}$ and $q = \gamma g$. This form has the general solution: $r_{\mathbb{S}} e^{\int p dt} = \int q e^{\int p dt} dt + C_r$. Solving the integration with respective p and q expressions gives (with $g = f(\theta)$):

$$r_{\mathbb{S}}(t) = f(\theta(t)) + C_r f(\theta(t)) e^{-\gamma t},\ C_r = \frac{r_{\mathbb{S}}(0)}{f(\theta(0))} - 1 \tag{4.37}$$

The analytical solution shows that the oscillator converges to the desired limit cycle behavior $\Xi_{\mathbb{S}}(\theta) = f(\theta)$ when $t \to \infty$. The convergence behavior is an exponential decay which is shaped by f. Now if we use a custom convergence function h instead of f to define the convergence behavior, we can modify Equation (4.37) to represent the new desired solution:

$$r_{\mathbb{S}}(t) = f(\theta(t)) + C_r h(t) e^{-\gamma t},\ C_r = \frac{r_{\mathbb{S}}(0) - f(\theta(0))}{h(0)} \tag{4.38}$$

To obtain the dynamical system yielding this desired solution we perform the steps done to obtain Equation (4.37) *backwards*. If we multiply both sides of Equation (4.38) with $\frac{e^{\gamma t}}{h}$ we have (again $g = f(\theta)$): $r_{\mathbb{S}} \frac{e^{\gamma t}}{h} = e^{\gamma t} \frac{g}{h} + C_r$, which can be rewritten as:

$$r_{\mathbb{S}} \frac{e^{\gamma t}}{h} = \int \frac{e^{\gamma t}}{h} \left(\gamma g + \frac{\dot{g}h - \dot{h}g}{h}\right) dt + C_r \tag{4.39}$$

Now if we define $e^p = \frac{e^{\gamma t}}{h}$ we obtain the general coefficients $p = \gamma - \frac{\dot{h}}{h}$ and $q = \gamma g + \frac{\dot{g}h - \dot{h}g}{h}$. With p and q defined, Equation (4.39) is a solution of a first order differential equation of the

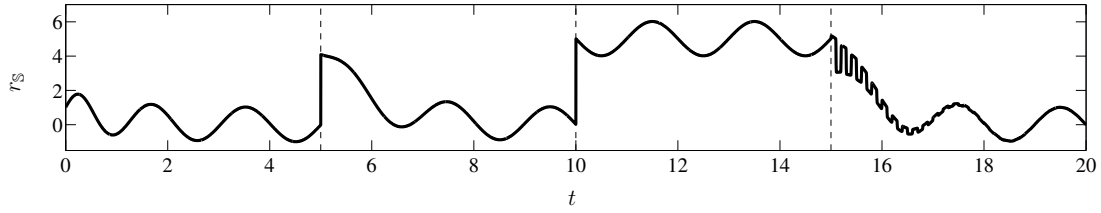


Figure 4.9: Arbitrary convergence behavior. This example shows the effect of different convergence behaviors h facing perturbations. For $t \in [0, 10)$, $h = 1$, for $t \in [10, 15[$, $h = e^{\gamma t}$, and for $t \in [15, 20[$, $h = \tanh(10\sin(10\theta)) + 1.5$. Perturbations are injected to the system at $t = 5$ and $t = 10$. As it is obvious, the perturbation at $t = 5$ is damped exponentially, and the one at $t = 10$ is not forgotten since the convergence behavior is canceled ($h = e^{\gamma t}$ so $\frac{\dot{h}}{h} = \gamma$). From $t = 15$ the system follows the smooth rectangular convergence defined by h. It is elegant that the steady state behavior does not depend on h (refer to Equation (4.38)).

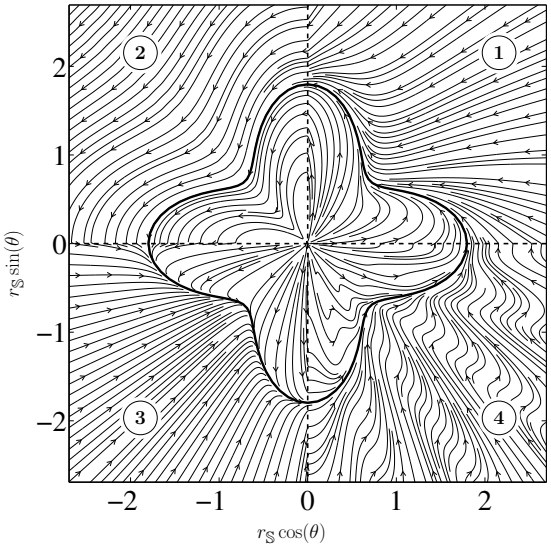


Figure 4.10: Arbitrary convergence behavior. For this example h changes in each quarter. In the first quarter $h = 1$ and exponential non-shaped convergence is obtained. The second quarter has $h = e^{\gamma t}$ which cancels the convergence behavior and the oscillator becomes neutrally stable. The third quarter is similar to the first quarter, with a 5 times bigger γ. Finally, for the fourth quarter $h = 2 + 0.5\sin(20\theta)$.

form defined in Equation (4.36). We obtain:

$$\dot{r}_{\mathbb{S}} + \left(\gamma - \frac{\dot{h}}{h}\right) r_{\mathbb{S}} = \gamma g + \frac{\dot{g}h - \dot{h}g}{h} \tag{4.40}$$

and we can simplify and write it as an ordinary differential equation for radius (with $g = f(\theta)$):

$$\boxed{\dot{r}_{\mathbb{S}} = \dot{f}(\theta) + \left(\gamma - \frac{\dot{h}(\theta, t)}{h(\theta, t)}\right)\left(f(\theta) - r_{\mathbb{S}}\right)} \tag{4.41}$$

Equation (4.41) along with the phase Equation (4.5) gives a first order morphed phase oscillator which the shape of its limit cycle is defined by f and the shape of its convergence by h. As

the desired solution in Equation (4.38) shows, the steady state solution does not depend on h and is only defined by f. The function $h(\theta, t)$ can be a function of phase, time, or both, and depends on the application. One can argue that the term $\gamma - \frac{\dot{h}(\theta,t)}{h(\theta,t)}$ can be generally written as $\gamma(\theta, t)$ which means having a phase - and/or time - dependent convergence rate. This is correct, however a representation like $\gamma(\theta, t)$ will not explain what the convergence behavior will be.

To ensure a stable system, the fixed-point $f(\theta) - r_{\mathbb{S}}$ in Equation (4.41) should be attractive. This means the condition $\gamma - \frac{\dot{h}(\theta,t)}{h(\theta,t)} \geq 0$ should be satisfied. Moreover, since $h(\theta, t)$ is in the denominator in Equation (4.41), $h(\theta, t)$ should not have a zero-crossing.

Figures 4.9 and 4.10 show examples of the arbitrary convergence behavior. Figure 4.9 depicts the effect of perturbation on different h functions. Figure 4.10 shows the phase portrait of a system with four different convergence behaviors. As it is clear, the limit cycle shape is not affected by the choice of the convergence behavior.

4.7 Learning

This section will explain how the scaling function f can be created from data points. We will additionally explain how the convergence behavior h can be fitted from data when Equation (4.41) is of interest. All the given descriptions are for one-dimensional cases, and extension to multi-dimensional cases is done by just repeating the same process for all dimensions and setting correct phase differences ϕ_{ij}.

4.7.1 Learning the Shaping Function

Let us assume that one dimensional input data is given as $\{t_i, y_i\}, i = 1 \dots N$, where t_i is the sample time, y_i is the desired output, and N is the number of data points. This data vector should represent a periodic activity. We first need to extract the frequency of oscillation, which methods like discrete Fourier transform or cross correlation can be used. After the frequency ϑ is determined, we create the phase data as $\theta_i = 2\pi\vartheta t_i$.

We can then use the values of y_i as the desired radius of the limit cycle. We add a constant offset δ_0 to y_i values in case they include negative values. This is due to the fact that f should be a positive function (other than the case where a base oscillator with a linear $\dot{r}_{\mathbb{B}}$ equation is used), and the same δ_0 should be subtracted when reading the output of the system. Consequently the data describing the scaling function f is defined as (with $\delta_0 > -\min_i(y_i)$):

$$f : \theta_i \longmapsto \frac{y_i + \delta_0}{\Xi_{\mathbb{B}}(\theta_i)},\ i = 1 \dots N \tag{4.42}$$

The dataset given in Equation (4.42) can be used to create any function approximator de-

scribing f, as long as it keeps the periodicity with a period of $1/\vartheta$. As recommended by [Ijspeert et al., 2013], the shaping function can be modeled with normalized weighted periodic Gaussian-like bases, known as von Mises basis functions:

$$f(\theta) = \frac{\sum_{k=1}^{K} w_k \psi_k(\theta)}{\sum_{k=1}^{K} \psi_k(\theta)}, \quad \psi_k(\theta) = e^{\frac{1}{\sigma_k}(\cos(\theta - c_k) - 1)} \tag{4.43}$$

where ψ_k is a von Mises basis centered at phase c_k and σ_k determines the span. If f is modeled so, then there are powerful tools like locally weighted regression [Schaal and Atkeson, 1998] to find the w_k parameters in an $O(KN)$ procedure. One additional benefit of using von Mises bases is that the resulting f function is smooth, i.e. it is C^∞ differentiable. This means that, even using the first order realization, all position, velocity, acceleration, etc states will be continuous, until perturbed.

We like to mention that one nice outcome of using a mixture of periodic basis functions to model f is that they can represent f in terms of simple movement/motor primitives, which can be linked to more biological explanation of how movements are coded and generated [Flash and Hochner, 2005, Mussa-Ivaldi et al., 1994, Thoroughman and Shadmehr, 2000]. Rhythmic Dynamical Movement Primitives [Ijspeert et al., 2013] are one example of such, and the framework here is a superset of rhythmic DMPs, and the notion of movement primitives apply as long as the f function is accordingly represented.

4.7.2 Learning the Convergence Function

In case where a morphed oscillator having a desired limit cycle shape but also with an explicit desired convergence behavior is of interest, one would use Equations (4.5,4.41) to model it. Let us assume that the given data $\{t_i, y_i\}, i = 1 \dots N$ describes the oscillation behavior from an initial condition y_0 which convergences to a periodic behavior[3]. We can simply take the last periods of oscillation (where the oscillator is already converged enough with respect to an error measure), and use this part to model f (Section 4.7.1). Knowing f, and by utilizing Equation (4.38), we have:

$$y_i + \delta_0 = f(\theta_i) + \frac{y_0 - f(\theta_0)}{h(0)} h(t) e^{-\gamma t} \tag{4.44}$$

[3] Our approach here is limited to having one single example. If multiple examples are given, one can first apply the process explained for one example, and then use the median of the resulting parameters.

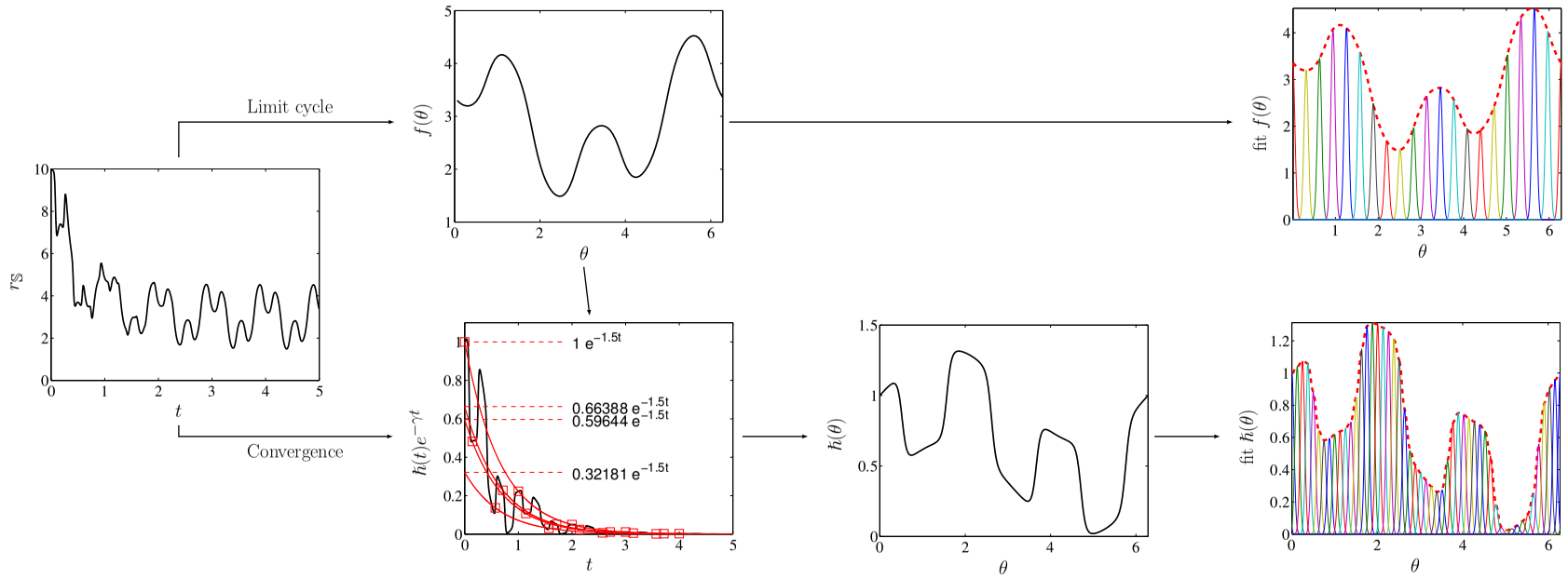


Figure 4.11: Learning the limit cycle and the convergence behaviors from an input data vector. The last cycle of the input data is used to extract the desired limit cycle and to build $f(\theta)$. Knowing $f(\theta)$, the convergence data is obtained by Equation (4.45), and is sampled at $2k\pi+\theta_o$ phases. The obtained sampled points are used to find an exponential fit. As the figure shows, different samples for different phase offsets θ_o result in the same convergence factor of $\gamma = 1.5$. After that the periodic part of the convergence function can be extracted. Both $f(\theta)$ and $\hbar(\theta)$ can be modeled with von Mises bases. This example uses 20 bases for $f(\theta)$ and 50 bases for $\hbar(\theta)$. The oscillator generating the input data for this example is described with $\gamma = 1.5$, $\Xi_S(\theta) = f(\theta) = 3+\arctan(\sin(3\theta+4))+\cos(\theta)$, and $h(\theta) = 2+\sin(\theta)+\tanh(3\cos(3\theta))$.

which can be rewritten as:

$$\hbar(t,\theta)e^{-\gamma t} = \frac{h(t)}{h(0)}e^{-\gamma t} = \frac{y_t + \delta_0 - f(\theta_i)}{y_0 + \delta_0 - f(\theta_0)} \tag{4.45}$$

where $\hbar(t,\theta)$ is the normalized convergence function. If $\hbar$ is desired to be a function of time, then γ can chosen arbitrarily and then $\hbar$ is numerically obtained, and a function approximation tool of choice can be used to model $\hbar(t)$.

The other case is where $\hbar$ is meant to be a periodic function of phase $\hbar(\theta)$. So the term $\frac{y_t+\delta_0-f(\theta_i)}{y_0+\delta_0-f(\theta_0)}$ should describe a pure periodic behavior multiplied by the exponential decay $e^{-\gamma t}$. This needs a correct estimation of γ. It is difficult, and can also be imprecise, to estimate the shape of $\hbar(\theta)$ and the convergence factor γ at the same time. The solution is to calculate γ without knowing the form of $\hbar(\theta)$. Since $\hbar(\theta)$ is a non-damped periodic function, its relative displacement in $2k\pi\vartheta + \theta_o, k \in \mathbb{Z}$ phases is zero. This means that if we sample $\hbar(t)e^{-\gamma t}$ in $k\vartheta + \frac{\theta_i}{2\pi\vartheta} + t_o$ time stamps, or $2k\pi + \theta_o$ phases, then the sampled data fits on a pure exponential decay $\alpha e^{-\gamma t}$ (t_o and θ_o are arbitrary offsets). Finally, γ can simply be estimated by fitting $\alpha e^{-\gamma t}$ on the newly sampled data, e.g. by least squares. After γ is estimated, data describing periodic phase-dependent function $\hbar$ is:

$$\hbar : \theta_i \mapsto \frac{y_t + \delta_0 - f(\theta_i)}{y_0 + \delta_0 - f(\theta_0)} e^{\gamma \frac{\theta_i}{2\pi\vartheta}} \tag{4.46}$$

and again von Mises bases with locally weighted regression can be used to model this data, like what was done for f. Figure 4.11 depicts the procedure to extract the limit cycle and convergence behavior from a given data vector.

4.7.3 Online Modulation

Different properties of the proposed morphed oscillators can be modulated online. Frequency modulation can be done by directly changing the ϑ values. This will have an immediate effect on the period of the system. Figure 4.12-top shows this property. Modulation of amplitude, offset and oscillation midpoint can be done by changing the f function on-the-fly. If the desired modulated output $\hat{\Xi}_{\mathbb{S}}(\theta)$ is:

$$\hat{\Xi}_{\mathbb{S}}(\theta) = a(\Xi_{\mathbb{S}}(\theta) - g) + g + o \tag{4.47}$$

where a is the amplitude magnification around midpoint g, and o is an added offset, then:

$$\hat{f}(\theta) = af(\theta) + \frac{(1-a)g + o}{\Xi_{\mathbb{B}}(\theta)} \tag{4.48}$$

where $\hat{f}(\theta)$ is the new scaling function giving the desired modulation. Examples are given in Figure 4.12. The effects of these modulations are not immediate, and act as swapping the limit

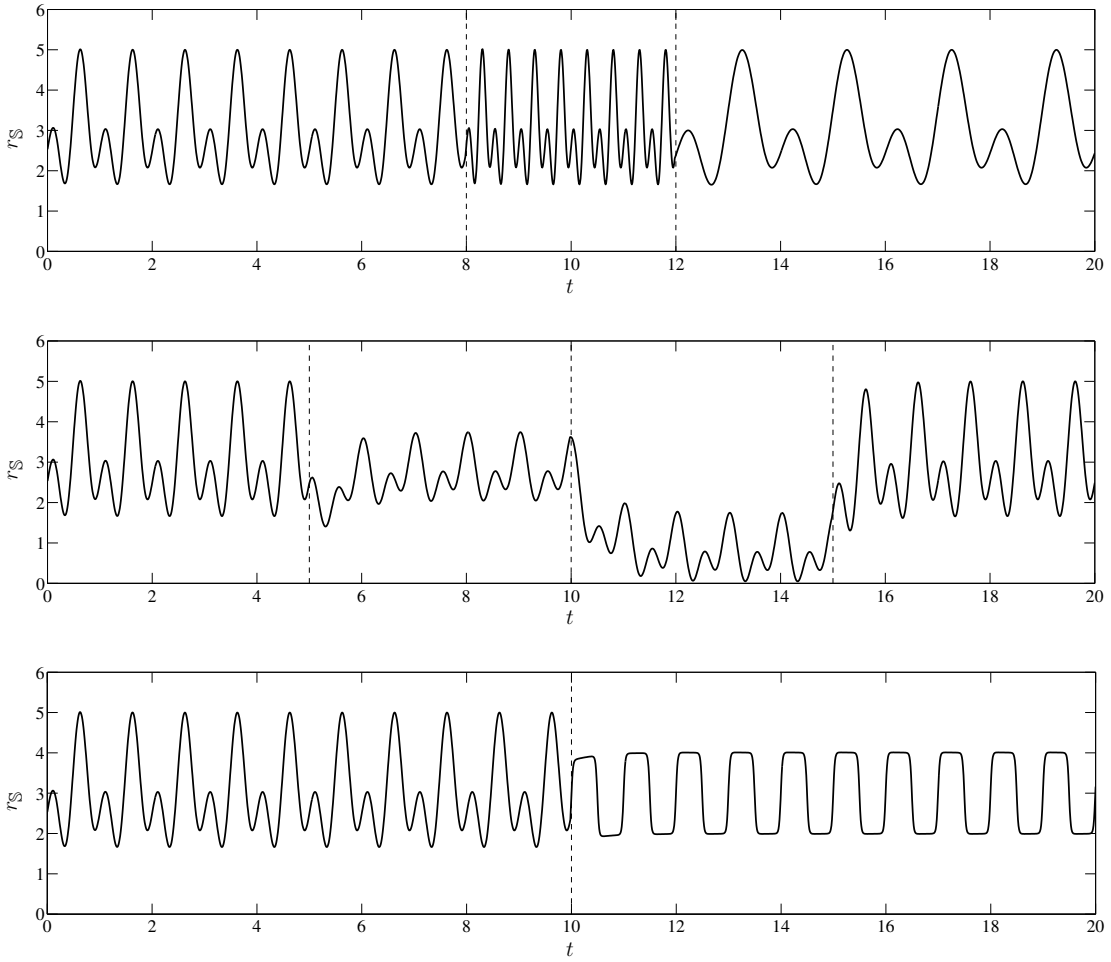


Figure 4.12: Online modulation of the limit cycle behavior. Top) modulation of frequency to $\vartheta = 2$ and then to $\vartheta = 0.5$. Middle) Modulation of amplitude by $a = 0.5$ around midpoint $g = 2.5$, then adding an offset of $o = -2$, and finally going back to the initial limit cycle. Bottom) Swapping the limit cycle with a new one. For all the figures $\gamma = 2$, and except the top figure, the value of γ determines the modulation/switching duration.

cycle of the system with a new one, so the system will gradually converge to the modulated limit cycle. All the above holds when f is replaced with a completely new $\hat{f}$ which can have a different shape than f, as the bottom plot in Figure 4.12 shows.

We like to mention that since the oscillators obtained by the introduced methodology are phase oscillators, frequency adaptation rule can easily be applied to them. With reference to [Righetti et al., 2006], a frequency adaptation rule for entrainment [Pikovsky et al., 2003] with an external periodic input can be simply written as:

$$\dot{\omega} = -\eta I(t) \sin(\theta) \tag{4.49}$$

where $I(t)$ is the external periodic signal and η is the adaptation rate. This means that instead

of having a constant ω in the phase equation $\dot{\theta} = \omega$, an adaptive dynamics is applied to $\dot{\omega}$. Please refer to [Righetti et al., 2006, Buchli et al., 2006a] for details.

4.8 Application

Morphed oscillators can be applied to different problems when there is a need to encode a desired periodic pattern as a limit cycle of a dynamical system. Moreover, morphed oscillators can be designed to be globally asymptotically stable, and have a desired convergence behavior, which advocates applications requiring state feedback integration. The concept of the morphed oscillators is new, so the list applications that we present is a combination of recently implemented ones, and propositions:

Locomotion: As we know from animals, the periodic joint angle profiles during different stages of locomotion are nonlinear and can be rather complex. The reader can refer to [Miller et al., 2013] for a sample human joint angle profiles, and to [Hasler et al., 1997] for a sample feline joint angle profiles. It is possible to use a simple phase generating system and apply output functions to directly obtain desired control signals (like $r_{\mathbb{S}} = f(\theta)$). This can be sufficient in cases where no feedback signal should be fed back to the oscillator (e.g. walking on a flat surface). However, as soon as there are feedback signals (e.g. for walking on irregular terrain), using an output function can lead to output discontinuities or non-smoothness ($r_{\mathbb{S}} = f(\theta) + e$ becomes discontinuous or non-smooth if e is discontinuous or non-smooth). Morphed oscillators provide a systematic way to *encode* the desired joint angles profiles in oscillator dynamics with known stability properties. This enables the control engineer to design feedback signals without the worry of pattern generation discontinuity or instability. We will see more on this topic in the next chapters.

Imitation: Morphed oscillators can be used for imitation of periodic tasks. In imitation, the tutor's demonstration can be of arbitrary complexity in terms of the trajectory shape. [Ajallooeian et al., 2012, Ijspeert et al., 2002a,b, Gams et al., 2008] give several examples of periodic tasks that can be imitated with a robot, including drumming. As discussed above with locomotion, such periodic tasks can be done using phase generating systems and output functions. In case of imitation however, there are two issues using output functions: 1) Like in locomotion, adding feedback tends to output discontinuity or non-smoothness. For example in the drumming task, if the drums are repositioned, there needs to be feedback signals to adapt for proper contact. 2) Imitation involves switching between different motor tasks, and using output functions can result in discontinuities when switching. Using morphed oscillators, one can code the imitated trajectories in dynamical systems which allow for feedback integration and also make the task transitions smooth (e.g. like the limit cycle modulation in Figure 4.12).

Neurorobotics: Coupled morphed oscillators can be used to create high-level computational

Central Pattern Generator (CPG) models [Matsuoka, 1987, Ijspeert et al., 2007, Cohen et al., 1982]. In this case, an oscillator represents the activity of a complete oscillatory center (instead of a single neuron or a small circuit) [Ijspeert, 2008]. The study in [Ijspeert et al., 2007] shows how coupled oscillators can be used to describe swimming, walking and the transition between them in salamanders, and applies those oscillators to the control of a salamander robot. The aforementioned study uses sine-wave outputs to control the spine joints. However recent recordings from salamanders [Karakasiliotis et al., 2013] show that spine movements are not purely sine-wave and have more nonlinearities. One can use the morphed oscillators instead of amplitude controlled oscillators in [Ijspeert et al., 2007] to create a more accurate high-level model of the salamander CPGs. Moreover, using morphed oscillators allows for encoding of limb joint angle trajectories as well, which are considerably more complex than spine trajectories [Karakasiliotis et al., 2013].

As another example, [Harischandra et al., 2010] uses time-driven pattern generators to activate a muscle model controlling the locomotion of a simulated salamander. The output patterns are used as abstract neural activation of the muscles. They only test their model for open-loop pattern generation. Morphed oscillators can be applied if one desires to equip such muscle pattern generators with feedback mechanisms. Morphed oscillators will then encode the activation pattern in stable oscillators which allow for smooth feedback integration.

Synchronization and assistance: Morphed oscillators give two-layer oscillators where the phase dynamics are decoupled from the radial dynamics. Having the decoupling, one can control radial dynamics limit cycle and convergence properties separately and leave the phases for eigenfrequency control. This motivates applications like model-free tracking in assistance and rehabilitation robotics, such as locomotion support using exoskeletons [Ronsse et al., 2010, 2011] or inter-agents phase synchronization [Mukovskiy et al., 2013] when the frequency coupling can be stably separated from radial dynamics.

Task initiation: As discussed in [Ernesti et al., 2012], "all periodic motions must be started in a nonperiodic way before the repeating pattern comes into play". It is important to be able to define the initiation trajectories of a periodic task, and the methods to do this with one single dynamical system are scarce [Ernesti et al., 2012]. As we showed in Section 4.6, morphed oscillators can be used to encode the desired initiation/convergence behavior into the same dynamical system that encodes the desired periodic task.

Stability analysis: The stability analysis given in Section 4.5 can be utilized to analyze the stability of a subset of complex systems. To give an example consider the following time-dependent system:

$$\dot{x} = -x\sin(t) + \left(1 - \frac{x^4}{e^{4\cos(t)}}\right)\left(\frac{2x}{e^{\cos(t)}} + \sin(\frac{x}{e^{\cos(t)}})\right)$$

It is not trivial to analyze the stability of this time-dependent system and describe its limit cycle properties (for example Matlab or Mathematica cannot solve this equation). However one can reformulate this system as an original morphed oscillator with $f(\theta) = e^{\cos(\theta)}$ and the base:

$$\dot{\theta} = 1, \quad \dot{r}_{\mathbb{B}} = (1 - r_{\mathbb{B}}^4)(2r_{\mathbb{B}} + sin(r_{\mathbb{B}}))$$

which has one stable fixed point at $r_{\mathbb{B}} = 1$ and an unstable one at $r_{\mathbb{B}} = 0$. This base satisfies the conditions in Equation (4.24) for $r_{\mathbb{B}} > 0$, and consequently, utilizing Equation (4.21), the system converges to the time driven limit cycle $e^{\cos(t)}$ for $x > 0$. The same analogy can be used to analyze the stability conditions of other similar dynamical systems if they can be rewritten into a morphed oscillator form. This method can be helpful in cases where classical stability analyses fall short.

□

One should keep in mind that morphed oscillators provide only the core building block for the above applications, and should be enriched with proper and task-specific feedback mechanisms. The role of the morphed oscillator is to encode the desired periodic trajectories into stable oscillators of desired order, which allows for feedback integration, smooth modulations and continuous transitions.

Many of the above applications can also be done applying rhythmic Dynamical Movement Primitives (DMP). So the question is why to design morphed oscillators. There are three answers to this question:

1. Rhythmic DMP is a 2nd order form of the morphed oscillators having a base with linear radial dynamics (see Section 4.3.1). Consequently what can be done with a rhythmic DMP can also be done with a morphed oscillator. Moreover, the idea of DMP builds upon the use of Gaussian like basis functions in order to represent the limit cycle shape *indirectly* via the forcing term, while the methodology of morphed oscillators allows for *direct* definition of the limit cycle shape using any desired tool for the representation of the shaping function (Gaussians, splines, etc).

2. Using the morphed oscillators, the system designer has the option to choose the order of the dynamical system. As we will see in the next chapters, tasks like locomotion *can* be done with 1st order morphed oscillators, which eliminates extra numerical integrations of a 2nd order system. On the other hand, [Nemec and Ude, 2012] shows that 3rd order dynamics can be useful for sequencing robotic tasks which need acceleration continuity. Moreover, if one look for jerk continuity in a periodic task, as in [Petrinec and Kovacic, 2007], then a 4th order system should be applied.

3. The methodology of morphed oscillators allows for a variety of different convergence behaviors. One can use different bases to obtain different levels of nonlinearity in the convergence. For example, using a base with linear radial dynamics (like DMP) results in simple exponential convergence, while using a base with polynomial radial dynamics can give a convergence behavior which is weak near the limit cycle and strong when far. One should keep in mind that the convergence behavior affects the external feedback effect, and having the choice of convergence behavior gives the system designer the freedom to properly choose the needed convergence.

4.9 Discussion

We presented a general methodology to morph a chosen phase oscillator, which acts as a base oscillator, to an oscillator with a desired limit cycle shape. The main idea of this methodology is based on a diffeomorphic phase-based scaling map which morphs the dynamics of the base to the desired one. The given methodology creates first order oscillators, and was extended to represent second order and $n-$th order oscillators. This realizes a general and populated family of nonlinear oscillators with any desired order.

Compared to the general approach of using recurrent neural networks to create nonlinear oscillators, using the presented methodology will reduce the design/training complexity. If one desires to obtain a desired limit cycle behavior out of a recurrent neural network, then he/she should employ rather complex training techniques like backpropagation through time [Rumelhart et al., 1986], and check for local asymptotic stability and unwanted local minima afterwards. The training technique in these approaches need to know about the internal dynamics of the network. However, for the morphed oscillators, the training process (to model the limit cycle shaping function f) only needs to know the limit cycle shape of the base oscillator, and not its transient dynamics, to be able to create the desired nonlinear oscillator. This gives the advantage that the training procedure gets reduced to a static function approximation.

Other than being an interesting mathematical challenge to create nonlinear oscillators with arbitrary limit cycle shapes, many motion control applications need such oscillators which makes this problem even more interesting. One very good example is the control of locomotion and use of nonlinear oscillators as pattern generators. This is widely known as the problem of designing Central Pattern Generator (CPG) [Ijspeert, 2008] models for locomotion. As Ijspeert mentions in [Ijspeert, 2008], to be able to systematically create CPGs, design of coupled nonlinear oscillator exhibiting desired limit cycles should be tackled. We believe that the approach here is general and systematic and helps to ease the design of CPGs. Moreover, since one can design globally asymptotically stable limit cycle systems with the introduced methodology, the inclusion of feedback signals will not affect the stability properties, and consequently broadens the types of feedback signals that can be included.

The nonlinear oscillators obtained by the introduced methodology are one dimensional phase

oscillators, and can get phase coupled to create a multidimensional system. This is different from a recurrent neural network which is multidimensional by design. The methodology here makes the creation of a multidimensional system easy by dividing it into low dimensional subsystems. However the radial dynamics of different dimensions are not directly coupled. This means that a perturbation on one dimension's radial state will not affect the other dimensions, other than when being explicitly coupled. This is different from a recurrent neural network where all the state dynamics are normally coupled, and this can be considered as an advantage or disadvantage depending on the application.

We also introduced the possibility to have an explicitly defined custom convergence behavior. This custom convergence behavior is apart from the choice of the limit cycle shape, and both arbitrary limit cycle shape and convergence can be obtained in one same dynamical system. This gives the possibility to include periodic tasks as well as their non-periodic initiation in one single system. This is useful in many rhythmic motor control tasks which need initiation, like locomotion. We have also explained how to learn both the limit cycle shape and the convergence behavior from given data, which makes this tool appropriate to be used for learning rhythmic tasks by imitation.

The stability analysis for the general family obtained from the introduced methodology was given. If one is looking for globally asymptotically stable limit cycle systems, the stability conditions can direct him/her to the choice of the base oscillators he/she has. More importantly, if one is forced to use a specific base (e.g. by implementation constraints), the given stability analysis can be used to know the stability bounds of the resulting system, and ensure a safe system design.

It is worth mentioning that morphed oscillators are the superset of the rhythmic DMPs, and this places the applications of DMPs within the scope of the morphed oscillators, with extended design choices.

In the end, we expect the introduced methodology to ease and systematize the design of nonlinear phase oscillators for different applications including robotics and motion control. Morphed oscillator are computationally light and simple to implement, which makes them appropriate tools for robotics application with real-time constraints. We are exploiting these features, and we will see in the next chapters how morphed oscillators can be applied to the problem of rough terrain locomotion.

5 Sensory Feedback

I wonder: "Positive Feedback" leads to perturbation magnification and system instability, and still we are looking for "Positive Feedback" from the others!

Publication

Parts of the material presented in this chapter is taken from:

- Mostafa Ajallooeian, Soha Pouya, Alexander Sproewitz, and Auke J Ijspeert. Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3321–3328. IEEE, 2013.
- Mostafa Ajallooeian, Sebastien Gay, Alexandre Tuleu, Alexander Sprowitz, and Auke J Ijspeert. Modular control of limit cycle locomotion over unperceived rough terrain. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3390–3397. IEEE, 2013.

The previous chapter discussed how morphed oscillators with arbitrary limit cycle shapes can be constructed in a systematic way. First order morphed oscillators are used in the rest of this thesis as the building blocks of the Central Pattern Generator module. We use one oscillator per active degree of freedom. Let the following be a general representation of a single first order morphed oscillator:

$$\begin{aligned} \dot{\theta}_i &= 2\pi\vartheta \\ \dot{r}_i &= \dot{r}_{\mathbb{S},i} = \mathscr{D}_{\mathbb{S},r}(f_i,.) \end{aligned} \tag{5.1}$$

with $\mathscr{D}_{\mathbb{S},r}(f_i,.)$ describing the radial dynamics of the morphed oscillator constructed by f_i. We choose f_i to represent the desired joint angle profile for the ith active degree of freedom.

Therefore $r_i = \int_0^t \dot{r}_i \, \mathrm{d}t$ is the joint angle command sent to the low-level controller to generate the torque command τ_i.

There are four possible ways to introduce feedback: phase feedback $\xi_{\theta,i}$, command feedback $\xi_{r,i}$, torque feedback $\tau_{fb,i}$, or feedback by altering the shaping function f_i:

$$\begin{aligned} \dot{\theta}_i &= 2\pi\vartheta + \xi_{\theta,i} \\ \dot{r}_i &= \mathscr{D}_{\mathbb{S},r}(f_i,.) + \xi_{r,i} \overset{r_i}{\Longrightarrow} \boxed{\text{Low-level Controller}} \overset{\tau_i}{\Longrightarrow} \tau_i + \tau_{fb,i} \end{aligned} \tag{5.2}$$

This thesis will not extensively address the phase feedback case, and there are several other studies discussing the effect of phase feedback [Gay et al., 2013, Owaki et al., 2013]. Feedback through the alternation of f_i is discussed when a change in the gait or the foot trajectory is needed. What we will mostly explain in this chapter is how to systematically design mechanisms to generate the $\xi_{r,i}$ or $\tau_{fb,i}$ feedback signals. Keep in mind that, quantity-wise, $\xi_{r,i}$ is angular velocity and $\tau_{fb,i}$ is torque, and we need mechanisms which generate such quantities.

This chapter explains how the posture control and reflex mechanisms are implemented. The results of applying these mechanisms to 3D quadrupedal locomotion are presented later in Chapters 7 and 8, as we first need to put everything together in a control architecture, detailed in Chapter 6.

5.1 Posture Control

This section explains how we can systematically design modules which generate $\xi_{r,i}$ or $\tau_{fb,i}$ signals accounting for the posture control task. We will first borrow concepts from Virtual Model Control (VMC), and show how torque feedback can be generated. However, as discussed in Chapter 3, we assume that the torque-control capability is not available. Thus the procedure to generate torque feedback is only to develop ideas towards a second model which works in absence of the torque-control mode. For that, we adapt ideas from VMC and Jacobian based inverse kinematics to generate angular velocity feedback signals $\xi_{r,i}$.

5.1.1 Elementaries

Leg Jacobian

Definition

Jacobian Matrix: Suppose function $\mathscr{F}$ which takes $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ as input, and outputs $\mathbf{y} = \{y_1, y_2, ..., y_n\}$. The $n \times m$ Jacobian matrix of $\mathscr{F}$ is defined as:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \tag{5.3}$$

and it can be easily shown (through partial derivation) that:

$$\dot{\mathbf{y}} = \mathbf{J}\dot{\mathbf{x}} \tag{5.4}$$

We utilize the concept of floating-based forward kinematics Jacobian for both of the approaches that will be used to generate posture control feedback. Assume the coordinate frame fixed to the world $\mathbf{o}_w$, and the floating coordinate frame fixed to a part of the robots body (e.g. foot) $\mathbf{o}_p$. Also take $\mathbf{q}$ as the joint angles of the robot, $\{x, y, z\}_m$ as the position of the main body (e.g. trunk) with respect to $\mathbf{o}_w$, and $\mathbf{R}_m$ as the rotation matrix describing the robot's main body orientation with respect to $\mathbf{o}_w$. The forward kinematics relation describing the position and the orientation of $\mathbf{o}_p$ in terms of $\mathbf{q}$, $\{x, y, z\}_\mathrm{m}$, and $\mathbf{R}_\mathrm{m}$ can be written as:

$$\begin{bmatrix} & x_p \\ \mathbf{R}_p & y_p \\ & z_p \\ \mathbf{0} & 1 \end{bmatrix} = \mathscr{F}(\mathbf{R}_m, x_m, y_m, z_m, \mathbf{q}) = \begin{bmatrix} & x_m \\ \mathbf{R}_m & y_m \\ & z_m \\ \mathbf{0} & 1 \end{bmatrix} \prod_{i}^{\substack{kin.\ chain \\ m \to p}} \mathscr{H}(q_i) \tag{5.5}$$

where $\mathscr{H}(q_i)$ is the local homogeneous transform [Sciavicco and Siciliano, 2000] of the joint q_i.

Throughout this thesis, we always have three active degrees of freedom per leg, and the output is the 3D position of the feet. So we have:

$$\mathbf{J}_l = \begin{bmatrix} \frac{\partial x_{p,l}}{\partial q_{\mathrm{AA},l}} & \frac{\partial x_{p,l}}{\partial q_{\mathrm{PR},l}} & \frac{\partial x_{p,l}}{\partial q_{\mathrm{FE},l}} \\ \frac{\partial y_{p,l}}{\partial q_{\mathrm{AA},l}} & \frac{\partial y_{p,l}}{\partial q_{\mathrm{PR},l}} & \frac{\partial y_{p,l}}{\partial q_{\mathrm{FE},l}} \\ \frac{\partial z_{p,l}}{\partial q_{\mathrm{AA},l}} & \frac{\partial z_{p,l}}{\partial q_{\mathrm{PR},l}} & \frac{\partial z_{p,l}}{\partial q_{\mathrm{FE},l}} \end{bmatrix}, \quad l = 1..4 \tag{5.6}$$

where $\{x, y, z\}_p$ refer to the foot position with respect to the world frame $\mathbf{o}_w$, and $q_{\mathrm{AA},l}$, $q_{\mathrm{PR},l}$ and

$q_{\text{FE},l}$ are the joint angles (Abduction/Adduction, Protraction/Retraction, Flexion/Extension) of the lth leg.

The Jacobian matrix in Equation 5.6 is a square matrix. Moreover, the range of motion for the legs of all our robots does not allow for singular configurations. This means that $\mathbf{J}_l$ will always be full-rank and invertible. It is important to note that the Jacobian matrix does not depend on $\{x, y, z\}_m$ (they get eliminated in the derivation).

Roll, Yaw, Pitch

Rigid body rotations can be described using different representations, including Euler angles, quaternions, axis-angle rotation, etc. We use Tait–Bryan angles (also known nautical angles or Cardan angles) to define roll (around x), yaw (around y), and pitch (around z), as shown in Figure 5.1.

Definition

Roll, yaw, pitch: Suppose that $\mathbf{R}$ is the column-major rotation matrix describing the rotation of the robot w.r.t. the world frame, then:

$$\psi_{roll} = \text{atan2}(\mathbf{R}_{[2,1]}, \mathbf{R}_{[2,2]}) \tag{5.7}$$

$$\psi_{yaw} = \text{atan2}(-\mathbf{R}_{[2,0]}, \sqrt{\mathbf{R}_{[2,1]}\mathbf{R}_{[2,1]} + \mathbf{R}_{[2,2]}\mathbf{R}_{[2,2]}}) \tag{5.8}$$

$$\psi_{pitch} = \text{atan2}(\mathbf{R}_{[1,0]}, \mathbf{R}_{[0,0]}) \tag{5.9}$$

where $\square_{[i,j]}$ provides element wise access to the element on ith row and jth column. Zero-based indexing is used here.

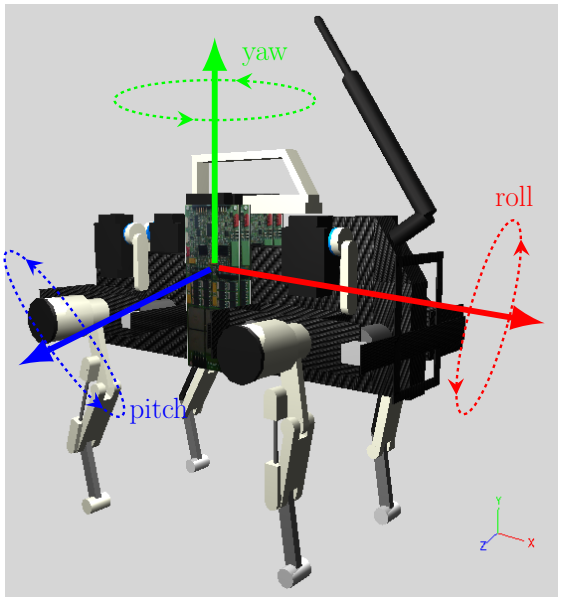


Figure 5.1: Definitions of roll (around x), yaw (around y), and pitch (around z).

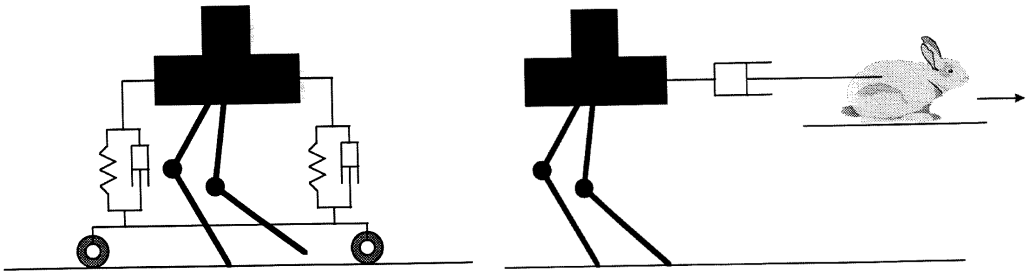


Figure 5.2: Two intuitive examples of VMC. *Left,* the granny walker which keeps the altitude controlled. *Right,* The bunny, which makes the robot instantaneously move forward. Pictures are taken from [Pratt et al., 2001].

Straight Rotation Matrix

It is well known that the measurement of the yaw angle, even utilizing high-end IMUs, includes time delays and drift. For all the hardware experiments and for slope estimation, we use the straight rotation matrix instead of the full rotation matrix. Straight rotation matrix is calculated by pre-multiplying the rotation matrix with a value which we call the safe yaw. Safe yaw is the angle of the projection of the robots heading vector on the frontal plane, and is calculated as:

$$\psi_{yaw,safe} = \text{atan2}(\mathbf{R}_{[2,0]}, \mathbf{R}_{[0,0]}) \tag{5.10}$$

with zero-based indexing used. The straight rotation matrix is:

$$\mathbf{R}_{str} = \mathscr{R}(0, -\psi_{yaw,safe}, 0)\mathbf{R} \tag{5.11}$$

where $\mathscr{R}(.,.,.)$ is the function to construct a rotation matrix from roll-yaw-pitch angles.

5.1.2 Virtual Model Control

The main idea of Virtual Model Control (VMC) is to attach virtual components to a robot, as if they had existed, and generate joint torques which simulate them [Pratt et al., 2001]. VMC "borrows ideas from virtual reality, hybrid position-force control, stiffness control, impedance control, and the operational space formulation" [Pratt et al., 2001]. Two rather interesting examples of virtual components used for locomotion, as described in [Pratt et al., 2001], are given in Figure 5.2.

The virtual components can be any force-generating element including springs and dampers. Suppose that a virtual element is attached to the point $\mathbf{o}_p$ on the robot, and generates the

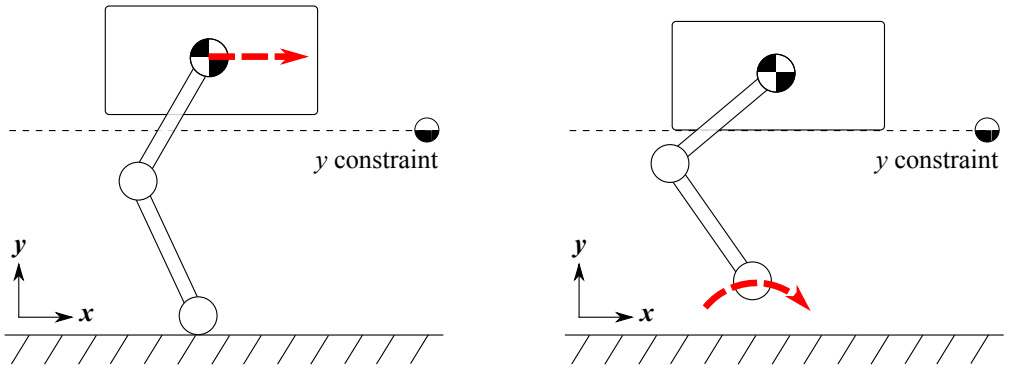


Figure 5.3: Simple constrained 1-leg model. A hard contact model is utilized. Trunk cannot rotate. Trunk y is constrained and cannot become smaller than a threshold. This means that the trunk will freely translate in stance (left), and the foot will be moving in the swing (right), where the trunk is stationed on the constraint line.

force $\mathbf{f}_{vm}$ with respect to a base frame $\mathbf{o}_b$. The principle of virtual work tells us that:

$$\boldsymbol{\tau}_{vm} = \mathbf{J}^{\top}\mathbf{f}_{vm} \tag{5.12}$$

where $\mathbf{J}$ is the Jacobian of the forward kinematics from $\mathbf{o}_b$ to $\mathbf{o}_p$ with respect to $\mathbf{o}_b$. Equation 5.12 is quite handy as it makes force control possible without depending on the mass and inertia properties of the system.

Simple Model

We aim to use concepts from VMC and apply them to quadrupedal posture control. Before that, we draw a simple example to show how to integrate VMC and CPG modules. Figure 5.3 depicts a simple constrained platform. The leg consists of two equally sized massless links. The only mass is a point mass centered in the trunk. The trunk cannot rotate and only translates in xy plane. The y position of the trunk is constrained by a virtual line and cannot fall below a predefined threshold. Trunk can lean on the constraint line, which causes the foot to move instead of trunk. To put it simple, the model behaves like a baby walker.

Let us assume that a CPG module is generating the joint trajectories leading to forward locomotion. These joint trajectories are then used to generate the joint torques $\boldsymbol{\tau}$ through the stance phase. The joints are position controlled in the swing phase as they are assumed to be massless. The behavior of this system is illustrated in Figure 5.5-left. Now assume the task of minimizing trunk y through stance, without modifying the CPG's encoded limit cycle. One can simply attach a virtual spring to the trunk along the y axis to pull the trunk down, as shown in Figure 5.4-*left*. These virtual spring, if they existed, would generate the virtual force $k(y_{const.} - y)$ which would pull the trunk down towards the constraint line. The effect of the

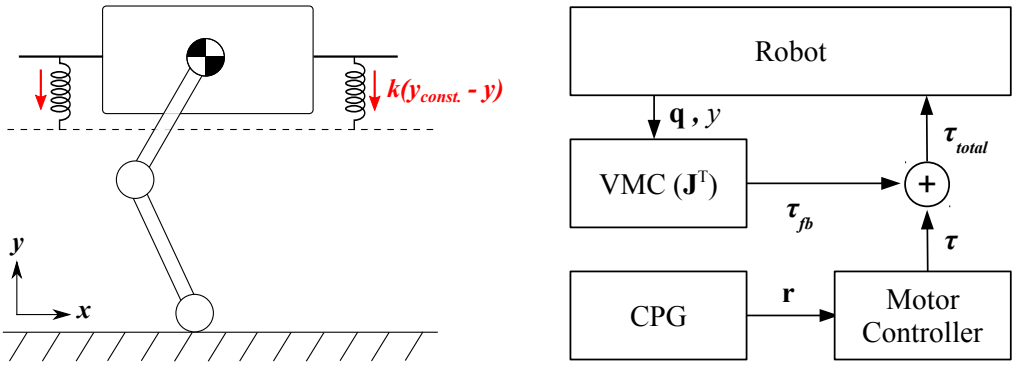


Figure 5.4: Simple trunk height control. *Left*, virtual springs to control trunk height. These springs do not physically exist, and they should be simulated through generating joint torques. Virtual springs generate the virtual force $k(y_{const.} - y)$, which is transformed into feedback torques using the Jacobian transpose. *Right*, the control diagram showing how CPG and VMC are integrated at the torque level. Note that CPG does not depend on sensory information. $\mathbf{q}$, the joint angles, are needed to calculate the Jacobian matrix at each timestep.

virtual springs can be simulated by:

$$\boldsymbol{\tau}_{fb} = \mathbf{J}^{\top} \begin{bmatrix} 0 \\ k(y_{const.} - y) \end{bmatrix} \tag{5.13}$$

where $\boldsymbol{\tau}_{fb}$ will be the torque feedback added to the CPG torque $\boldsymbol{\tau}$, see Figure 5.4-*right*.

Figure 5.5 illustrates the effect of the feedback torque with different feedback gains k. It is easy to see that the feedback always leads to a lower stance apex height. What is important to mention is that feedback can overshoot: trunk y can reach the $y_{const.}$ before the end of the stance phase. That is why a phase resetting mechanism is added to the CPG to transit to the swing phase as soon as y hits the constraint line. CPG's dynamics will then, at the beginning of the swing phase, make the joint position states converge to the desired profiles.

It is worth mentioning that the purpose of this example is not to provide a perfect solution to the stated problem, and is to show how the VMC feedback can provide task-specific adjustment torques additional to the CPG torques. One can add dampers along the introduced springs to have a smoother behavior and prevent the early transition to the swing phase.

3D Quadruped

Now that the basic concept of how VMC and CPG are integrated is clear, we discuss the virtual components used for an unconstrained 3D quadruped. Here we take a modular approach and use separate virtual components for different tasks, like in [Pratt et al., 2001]. We attach virtual springs to the robot to correct for body attitude, lateral skew, and direction during locomotion. The output of the posture controller is the total of the torques generated by these

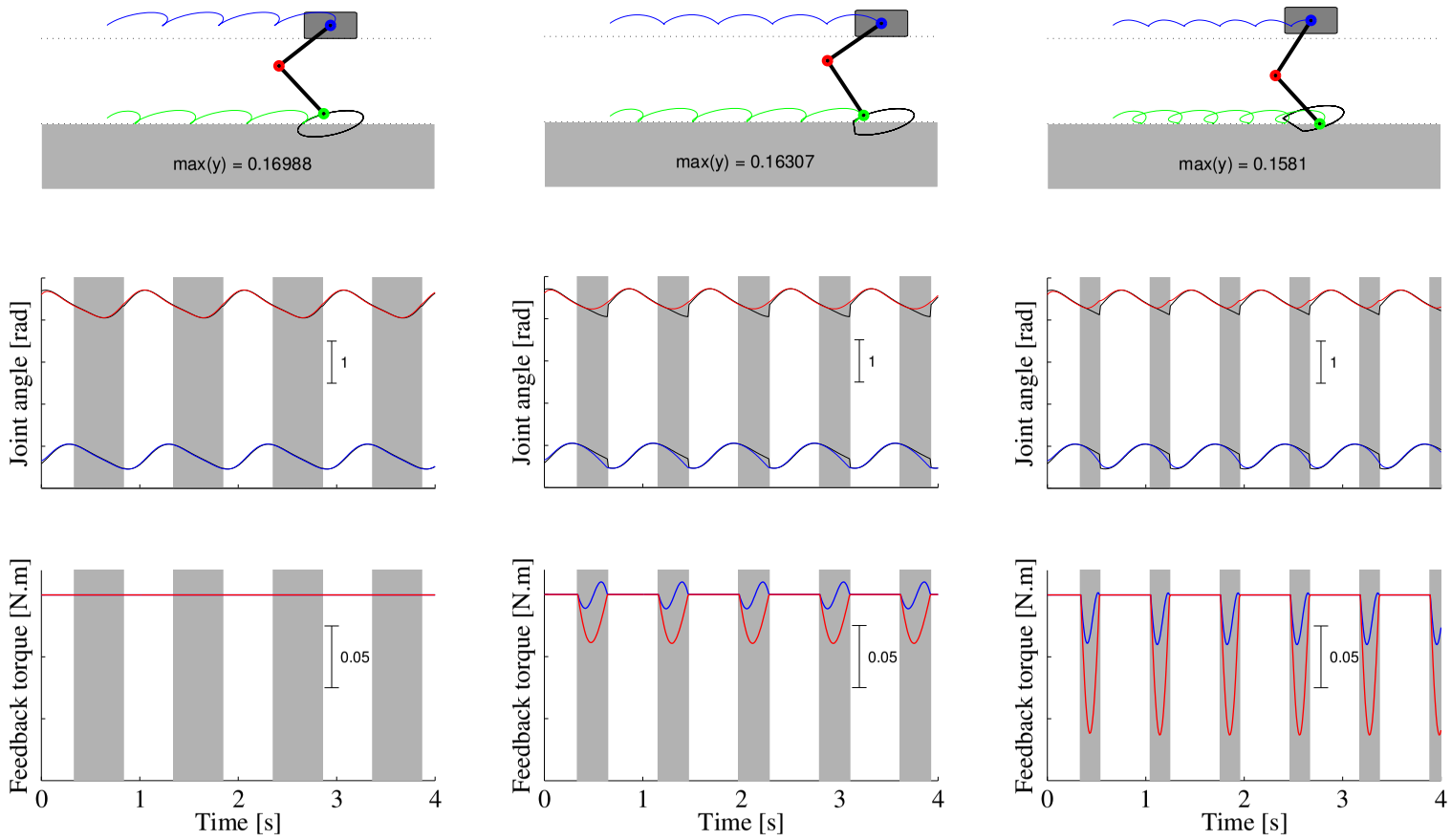


Figure 5.5: Simulation of the simplified model with different virtual model feedback gains. Blue lines are for the hip, red ones for the knee, and the clack ones are the references. *Left*, the open-loop case ($k = 0$). *Middle*, $k = 50$. Trunk y reaches the constraint line slightly before the open-loop swing onset, and the stance time is shortened by phase resetting. *Right*, $k = 200$. The stance time is largely shorter due to the fact that the virtual spring strongly pulls the body down and hits the constraint line rather quickly. It can be seen that the trunk height is minimized by the increase of the feedback gain.

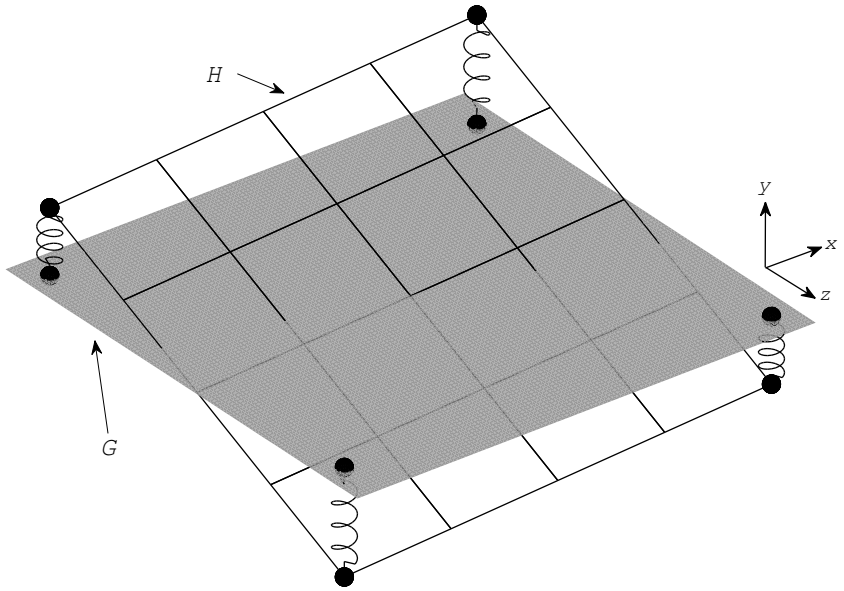


Figure 5.6: A schematic figure depicting the virtual springs for attitude control. A virtual plane, H, is attached to the trunk of the robot. Virtual springs are connected between H, and another virtual plane G lying parallel to the ground and passing through robot's trunk. There virtual springs generate virtual forces which will correct the robot's attitude.

three components:

$$\boldsymbol{\tau}_{fb} = \boldsymbol{\tau}_{att} + \boldsymbol{\tau}_{lat} + \boldsymbol{\tau}_{trn} \tag{5.14}$$

Attitude control: Assume a hypothetical plane connected to the center of trunk of the robot (H), and another plane passing through the center of the trunk lying horizontal w.r.t. world coordinates (G). As depicted in Figure 5.6, one can attach virtual springs between the corners of H and their vertical projections (w.r.t. world frame) on G. These virtual springs naturally generate forces which adjust the trunk attitude (both roll and pitch) to be parallel to the ground:

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix} = \mathbf{R} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.15}$$

$$\mathbf{f}_{att,l} = k_{att} \begin{bmatrix} & 0 & \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} & & \mathbf{p}_l \\ & 0 & \end{bmatrix} \tag{5.16}$$

where $\mathbf{R}$ is the rotation of the trunk frame (and the hip frames) w.r.t. the world coordinates, $\mathbf{P}$

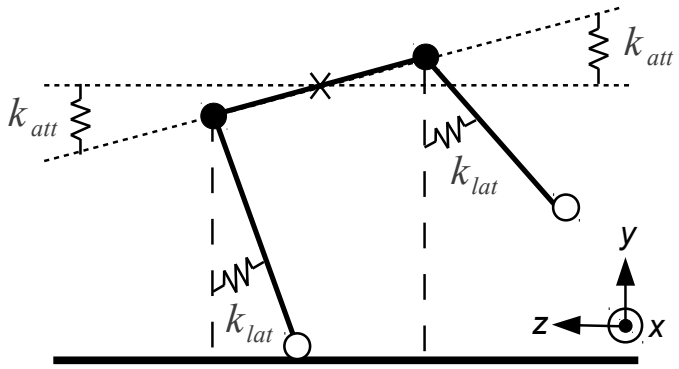


Figure 5.7: A front view sketch, and the virtual torsional springs for lateral skew correction. These spring are shown by k_{lat}, and the k_{att} springs are from Figure 5.6. The torsional springs directly generate torque, and there is no need for the Jacobian transpose in this case.

is the relative coordinates of the corners of H, k_{att} is the stiffness of the virtual springs and $\mathbf{f}_{att,l}$ is the vector of virtual forces for attitude adjustment, for the lth leg. Stance legs are used generate these forces. Each stance leg is used to generate forces generated by its corresponding virtual spring separately. If for example only two legs are on the ground, then they try to adjust the relative height of their hips, and the other two virtual forces are ignored.

Finally, if the force $\mathbf{f}_{att,l}$ is to be generated on the lth hip frame, it is the same as if the force $-\mathbf{f}_{att,l}$ is generated on the foot frame, and we have:

$$\boldsymbol{\tau}_{att,l} = -\mathbf{J}_l^\top \mathbf{f}_{att,l} \tag{5.17}$$

Lateral skew control: The attitude control mechanism keeps the roll and pitch contained, but does not address the cases where the robot is laterally skewed (a rhomboid-like posture). We continuously adjust the lateral skew by introducing virtual torsion springs at the hip abduction/adduction joints. The rest position of each virtual torsion spring is such that the corresponding leg is vertical (w.r.t. world frame) if that spring is at rest (Figure 5.7). So:

$$\boldsymbol{\tau}_{lat,l} = -k_{lat} \begin{bmatrix} \psi_{roll} - q_{l,AA} \\ 0 \\ 0 \end{bmatrix} \tag{5.18}$$

where k_{lat} is the stiffness of the torsional spring, ψ_{roll} is the trunk roll angle, and $q_{l,AA}$ is the hip AA joint angle for the lth leg.

Direction control: We implement a locomotion direction controller as virtual forces compensating for wrong heading direction. Based on the deviation of the robot from the desired direction, we generate sideways virtual forces with opposing signs in front and back of the

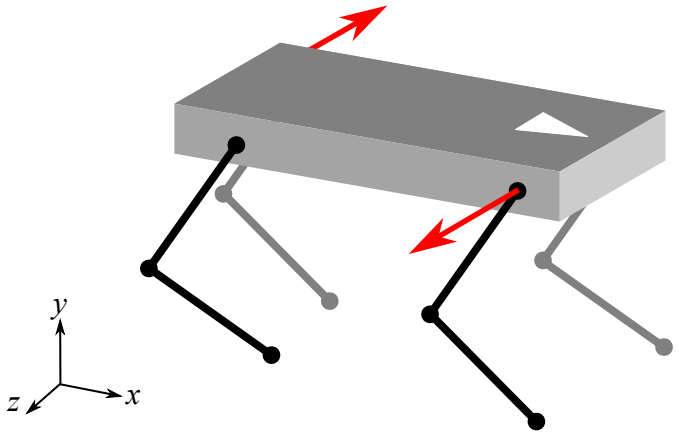


Figure 5.8: Virtual forces for turning. If the robot intends to turn right, a rightward virtual force is applied in the front, and a leftward force in the back. Keep in mind that these forces are only generated by the stance legs, and the swing leg do not contribute to generating these forces.

robot (legs are ordered as left-fore, right-fore, left-hind, right-hind, with zero-based indexing):

$$\mathbf{f}_{trn,l} = (-1)^{\lfloor \frac{l}{2} \rfloor} k_{trn} \begin{bmatrix} \sin(\Delta\psi_{yaw}) \\ 0 \\ \cos(\Delta\psi_{yaw}) \end{bmatrix} \tag{5.19}$$

where k_{trn} is the turning gain, and $\Delta\psi_{yaw}$ is the difference between the desired heading angle and the current one. Finally Equation 5.17 is used to calculate the feedback torque $\boldsymbol{\tau}_{trn}$ from the virtual forces $\mathbf{f}_{trn,l}$. Figure 5.8 depicts an example situation and the generated virtual forces.

Wrap-up

We presented a modular approach to the posture control problem borrowing concepts from VMC. Virtual springs are attached to the robot's main body, which generate virtual forces accounting for robot's attitude, lateral posture, and heading. These virtual forces are converted to joint torque feedback signals using the transpose of the feet position Jacobian matrix.

Here we have violated one of the assumptions that we mentioned as a part of the problem statement in Chapter 3: unavailability of torque-control mode. We can of course exploit the torque-control capability in simulation, and the described posture control method is used to control a simulated quadruped which is presented later in Chapter 7, Section 7.1. However, the posture control based on VMC cannot be ported to our hardware robot as it does not have torque-control capability. Based on the insight acquired from the presented posture controller, we design a second posture controller which does not depend on torque-control capability, as we will see next.

5.1.3 Virtual Velocities

As explained in the previous section, a modular posture controller can be implemented using concepts from VMC. However, there are two points of concern regarding the proposed approach:

1. VMC needs the force/torque control possibility to work; and
2. Task modularity in the posture control, i.e. having different elements for attitude, lateral and direction control, means that they can generate opposing forces not satisfying all the tasks. Nevertheless, modularity can be useful such that bigger gains can be set for more important tasks, but more parameters need to be tuned for control.

Based on these points, this section proposes a whole-body 1-module posture controller which does not depend on the force/torque control capability.

As mentioned in Equation 5.2, one way to introduce feedback is through adding joint velocities $\boldsymbol{\xi}_r$ to the output dynamics. We divide $\boldsymbol{\xi}_r$ into two parts: 1) $\boldsymbol{\xi}_{pos}$ for posture control (this section), and $\boldsymbol{\xi}_{rfx}$ for reflexes defined in Section 5.2.

By the definition in Equation 5.4 we know that joint space velocities can be generated from task space velocities. We exploit this fact to design angular velocity feedback signals for posture control based on task-space velocities:

$$\boldsymbol{\xi}_{pos} = \mathbf{J}^{-1}\mathbf{v} \tag{5.20}$$

where $\mathbf{v}$ is the vector of task space velocities to perform the desired posture control task. Hence, posture control can be reduced to the design of the task-space velocity vectors.

Simple Model

We explore a simple question here: "What happens if the virtual forces generated by virtual elements of VMC are assumed to be virtual velocity vectors (with a proper change of units and dimensions)?" To explore the answer, we recall the simple constrained 1-leg model in Figure 5.3. The task is still to reduce the trunk y though the stance phase. We adapt Equation 5.13 by assuming that the virtual forces are virtual velocities, so:

$$\boldsymbol{\xi}_{pos} = \mathbf{J}^{-1}\begin{bmatrix} 0 \\ k(y_{const.} - y) \end{bmatrix} \tag{5.21}$$

Figure 5.9 illustrates the behavior of the simple model under the feedback generated from Equation 5.21. The plot to the left shows the behavior of the system when there is not feedback, and the plots in the middle and right show the behavior for $k = 10$ and $k = 50$. It is easy to

Figure 5.9: Simulation of the simplified model with different virtual velocity gains. Blue lines are for the hip, red ones for the knee, and the black ones are the references. *Left*, the open-loop case ($k = 0$). *Middle*, $k = 10$. Trunk height is reduced, and no phase resetting was activated. *Right*, $k = 200$. The trunk height is largely reduced. The CPG dynamics is altered strongly in the stance phase, and the output signals smoothly converges back to the limit cycle by the initiation of the swing phase.

see that the task of minimizing trunk y is being properly done. We also observe that the joint angle command smoothly converges back to the desired profile when the stance phase is over and there is no feedback anymore. There is one man difference between the results here and the ones shown in Figure 5.5: The feedback does not overshoot, and no phase resetting is activated. The feedback generated from the virtual velocities is of first-order dynamics, and a first order attractor cannot overshoot. This is different from a virtual spring which is of second-order dynamics. Please note that the assumptions and approach taken here has similarities to resolved motion rate control [Whitney, 1969].

3D Quadruped

We implement a whole-body posture controller using the concept of virtual velocities. To have a general idea, Figure 5.10 illustrates how task space virtual velocities can be generated to adjust the posture.[1] If Figure 5.10-*top* is the present state of the robot, and the (arbitrary) desired body position and orientation of the trunk are the ones in Figure 5.10-*middle*, then virtual velocities in the Figure 5.10-*bottom* (red arrows) can be generated to adjust the posture while keeping the feet at the place they are (without creating slippage caused by the violation of the internal kinematic constraints). This is similar to adjusting the trunk posture in the null space of the feet positions Jacobian, but represented in a geometric manner.

The first step to the whole-body posture control is to estimate the ground inclination. This is explained later in Chapter 6 where the slope estimator module is described. Knowing the ground inclination, we try to keep the trunk pitch parallel to the local ground[2], and compensate for all the body roll, and at the same time have the desired yaw rotation (heading direction) applied. So the target rotation matrix is:

$$\mathbf{R}_{des} = \mathscr{R}\left(0, \psi_{yaw,des}, \psi_{pitch,gnd}\right) \tag{5.22}$$

where $\mathscr{R}(.,.,.)$ is the function to construct a rotation matrix from roll-yaw-pitch angles, $\psi_{yaw,des}$ is the desired yaw angle relative to the current heading, and $\psi_{pitch,gnd}$ is the estimated ground pitch.

Additionally, we want the vertical projection of the neck/tail point to be between the fore/hind feet, to prevent a laterally skewed posture. So the position adjustments are:

$$\mathbf{p}_{l,adj} = \frac{1}{2}(\mathbf{p}_l + \mathbf{p}_{contra\{l\}}) \ , \ l = 1..4 \tag{5.23}$$

where $\mathbf{p}_l$ and $\mathbf{p}_{contra\{l\}}$ respectively are the Cartesian position of the lth foot and its contralateral pair w.r.t. the frame attached to the robot's trunk.

[1] For clarity, a 2D illustration is given in Figure 5.10, and the same concept is applicable to a 3D model.
[2] Based on personal observation in dogs.

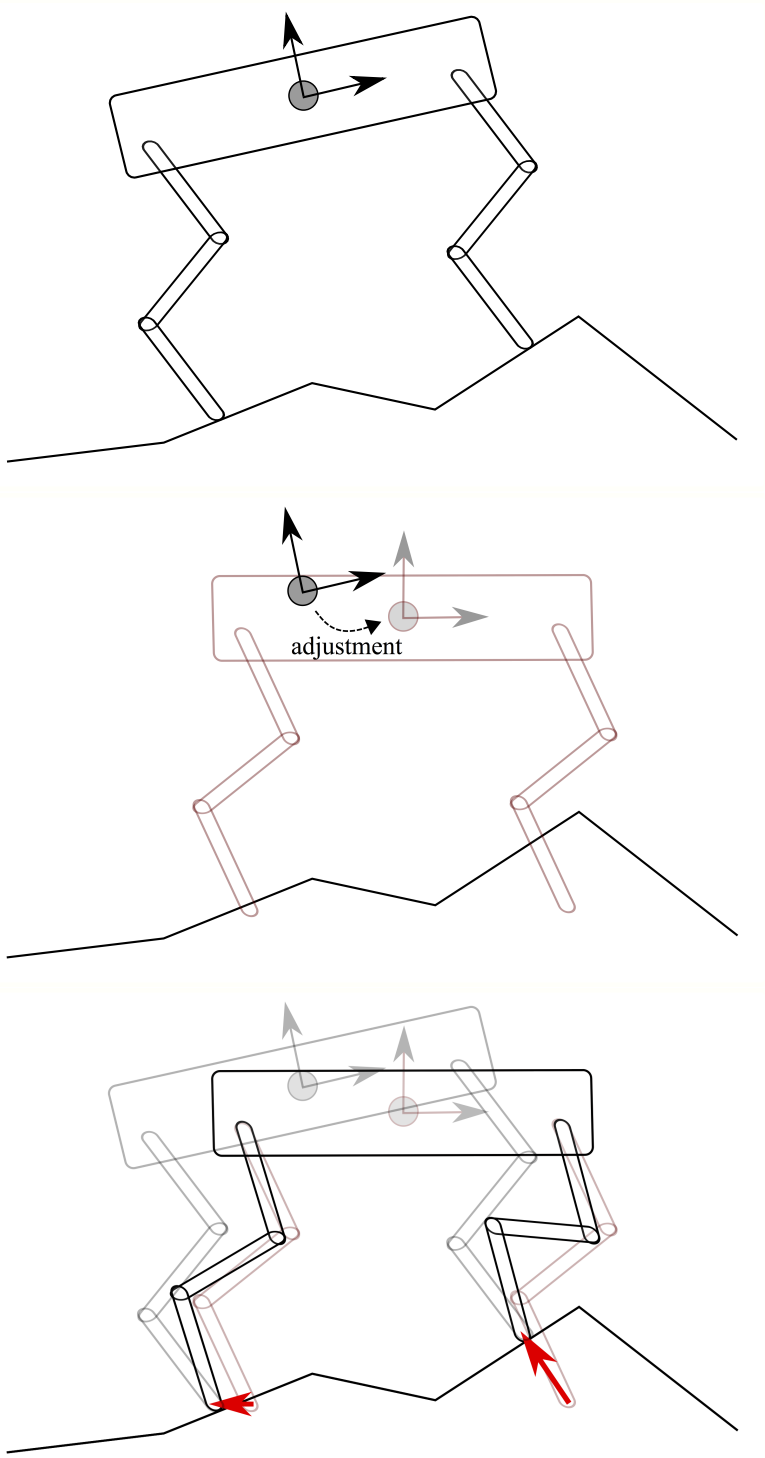


Figure 5.10: Whole-body posture control through the application of virtual velocities. *Top*, the current posture of the robot. *Middle*, a desired arbitrary adjustment of the trunk position and orientation. *Bottom*, The virtual velocities (red arrows) needed to be generated to perform the posture adjustment. The virtual velocities are the difference between the feet position in the top and middle figures (w.r.t. the world frame).

Finally, if $\mathbf{R}_{des}$ is the desired orientation, and $\mathbf{p}_{l,adj}$ the position adjustments that should be made, the task space virtual velocities performing this adjustments on the trunk are:

$$\mathbf{v}_l = \mathbf{R}_{des}(\mathbf{p}_l + \mathbf{p}_{l,adj}) - \mathbf{R}\mathbf{p}_l \ , \ l = 1..4 \tag{5.24}$$

and the required joint space velocity feedbacks are:

$$\boldsymbol{\xi}_{pos,l} = -k_{pos}\, \mathbf{J}_l^{-1}\mathbf{v}_l \ , \ l = 1..4 \tag{5.25}$$

The negative sign is due to the fact that the posture adjustments are defined for the trunk, but the Jacobians are written for the feet.

Wrap-up

We explained in this section how virtual forces can be replaced by virtual velocities, and how they can be utilized to design a whole-body posture controller for a 3D quadruped, without any need for torque-control. All the posture subtasks, i.e. attitude, lateral posture, and heading, are adjusted simultaneously through task-space virtual velocities. These virtual velocities are converted to joint velocity feedback signals using the inverse of the feet positions Jacobian matrices. The results of the application of this kind of posture control is presented in detail in Chapter 7, Section 7.2.

5.1.4 Angle-of-attack Control

We know from both the Raibert's control [Raibert et al., 1986], and the studies on the Spring Loaded Inverted Pendulum (SLIP) [Holmes et al., 2006], that the angle-of-attack can be chosen to accelerate or decelerate the body. A more vertical angle-of-attack will speed up the locomotion, while a more flat angle-of-attack will causes a break [Hodgins and Raibert, 1991]. We use this fact to change the angle-of-attack while locomoting on slopes or stairs, which needs adding (for upwards slopes) or removing (for downward slopes) energy to/from the system.

We implement two different ways of adjusting the angle-of-attack: 1) adding hip PR (Protraction/Retraction) offsets; or 2) shifting the foot trajectory. Both of these approaches are implemented by modifying the limit cycle shape of the CPGs, $f_i(.)$, on-the-fly as the change in the ground inclination is sensed.

Adding Hip PR Offset

The hip PR joint of the lth leg is controlled by a morphed oscillator, and its limit cycle shape is defined by $f_{l,PR}$. One can change the angle-of-attack by directly adding an offset to the hip PR

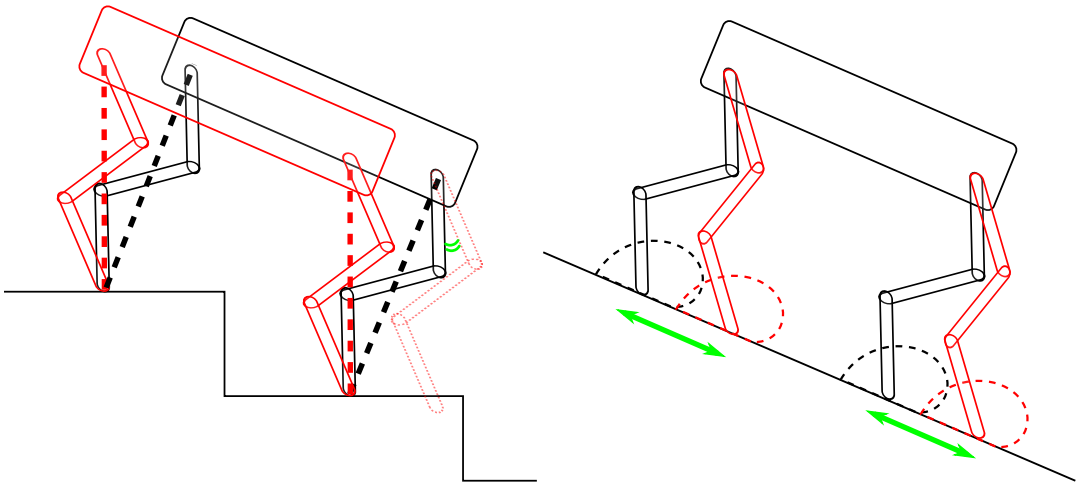


Figure 5.11: Angle-of-attack control strategies. A black figure is the initial posture, the red one is the posture after the strategy is applied, and the green marker indicates the change. *Left,* Adding an offset to hip PR. This strategy only affects the hip PR joint, and rotates the foot trajectory to be parallel to the ground. This strategy is good for the case of stairs. *Right,* Shifting the foot trajectory. This strategy needs the inverse kinematics of the foot trajectory to be recalculated. This approach is good for the case of slopes. It is important to mention that the robot cannot differentiate between slopes and stairs, and one strategy should be chosen.

joint's limit cycle (Figure 5.11-*left*):

$$f_{l,PR} \leftarrow f_{l,PR} + o_{PR}, \quad l = 1..4 \tag{5.26}$$

where o_{PR} is the added offset (same for all the legs). Note that by adding the hip PR offset o_{PR} the foot trajectory rotates by o_{PR} [rad] around the hip PR axis. We use a linear gain to correlate o_{PR} with the ground inclination:

$$o_{PR} = -k_s \psi_{pitch,gnd} \tag{5.27}$$

where $\psi_{pitch,gnd}$ is the estimated ground pitch (in Sagittal plane), and k_s is the activation gain.

For the case of locomotion over stairs, if $k_s = 1$, then the foot trajectory becomes parallel to the ground. For locomotion on a downward slope, since the foot trajectory is rotated, the contact will be late, and a leg extension reflex (later in section 5.2) will be activated to facilitate ground contact.

Please note that the estimation of the ground inclination, as explained later in Chapter 6, does not tell if the robot is on stairs or slopes. The only way to differentiate between stairs and slopes is to use 3D force sensors, which, as mentioned in Chapter 3, are not available for our case.

Shifting the Foot Trajectory

Changing the angle-of-attack can also be obtained through shifting the foot trajectory, as illustrated in Figure 5.11-*right*. If the default foot trajectory is defined as $\begin{bmatrix} u_x(.) \\ u_y(.) \end{bmatrix}$, then:

$$\begin{bmatrix} f_{l,PR} \\ f_{l,FE} \end{bmatrix} \leftarrow \mathscr{I}(\begin{bmatrix} u_x(.) + o_{FT} \\ u_y(.) \end{bmatrix}), \quad l = 1..4 \tag{5.28}$$

$$o_{FT} = -k_s \psi_{pitch,gnd} \tag{5.29}$$

where $\mathscr{I}$ is the 2D closed-form inverse kinematics of the foot position in the hip frame, o_{FT} is the shift in the foot trajectory, and k_s is the activation gain. Note that shifting the foot trajectory affects both the hip PR and knee FE limit cycle shapes as the joint level. It should be considered that with large values of k_s, the foot trajectory can be shifted outside the reachable working space of the leg.

This strategy is good for locomotion on slopes, and does not need any additional reflex to be activated. For the case of stairs, since this strategy does not rotate the foot trajectory, posture control should help to keep the body upright.

5.2 Reflexes

The previous section discussed how to design a posture control module. The posture control module is responsible for continuous adjustments of the posture, but is not designed to react to the events which need a rapid correction. Here we employ the concept of reflexes and design simple feedback mechanisms for quick and momentary adjustments.

By definition, a reflex is "an action or movement of the body that happens automatically as a reaction to something" [mer, 2014b]. Similarly, in biology, a reflex is an involuntary and almost instant movement caused by a stimulus [Purves, 2004]. Known examples of reflexes in humans include knee-jerk reflex, vestibulo-ocular reflex, etc. In the same context, there are also preflexes [Loeb, 1995]. Preflexes, different from reflexes, do not go though neuronal circuits and are rather "zero-delay" intrinsic feedbacks generated by the musculoskeletal system. In the context of this thesis, the term reflex is loosely tied to biology, and we call any quick and momentary reaction to an external stimulus a reflex.

Definition

Reflex: A quick and momentary reaction, through sensory feedback to the CPG, which is caused by an external stimulus.

Reflexes are crucial in cases where an unexpected event happens, and fast corrections are needed to prevent failure. There are three kind of reflexes that we address in this thesis, as detailed in the following.

5.2.1 Stumbling Correction Reflex

During locomotion on irregular terrain, legs can hit into obstacles while swinging forward. This can cause the robot to stumble and fall, or to aggressively change direction. Stumbling Correction Reflex (SCR) provides a simple mechanism to prevent stumbling while swinging the leg forward.

Forssberg et al. performed experiments on intact house cats to better understand the stumbling correction reflex. As explained in [Forssberg, 1979], stumbling corrective reaction is a phase-dependent compensatory reaction during locomotion. For the experiments, they used tactile stimuli to the paw aimed at the dorsum. They show that if the stimulus occurs during the swing phase, a short-latency activation of the flexor muscles inducing an additional flexion of the limb happens and helps lifting the paw over the obstacle.

We formulate SCR as a decaying impulse feedback to quickly flex the knee:

$$\begin{cases} \xi_{rfx,l,FE} \leftarrow k_{rfx} & \text{if stumbling} \\ \dot{\xi}_{rfx,l,FE} = -\alpha_{rfx}\xi_{rfx} & \text{else} \end{cases} \tag{5.30}$$

where α_{rfx} is the reflex decay rate, and k_{rfx} is the activation gain. As Figure 5.12-left illustrates, SCR causes the leg to flex, and provide the chance the lift over the obstacle. We mainly work with robots with three-segmented pantograph legs (as in the Figure 5.12), and knee/ankle flexion causes the foot to slightly move back. In case of two-segmented legs, we additionally activate the feedback, with a smaller amplitude, for the hip PR joint.

The aforementioned formulation is not the only way to implement SCR, and one other way is to directly modify the foot trajectory and add an extra arc to go over the obstacle while stumbling, as demonstrated in [Focchi et al., 2013].

5.2.2 Leg Extension Reflex

The study by Daley et al. [Daley and Biewener, 2006, Daley et al., 2006] shows that if a guinea fowl misses a contact at the beginning of the stance phase, then the leg is extended or at least kept extended until a contact is sensed, and they discuss that such a reaction stabilizes the locomotion over unperceived rough terrain. The Leg Extension Reflex (LER) can be implemented by extra extension of the knee joint when the expected contact is missing. LER is illustrated Figure 5.12-right, and implemented the same way as in Equation 5.30, only with a negative sign for k_{rfx}.

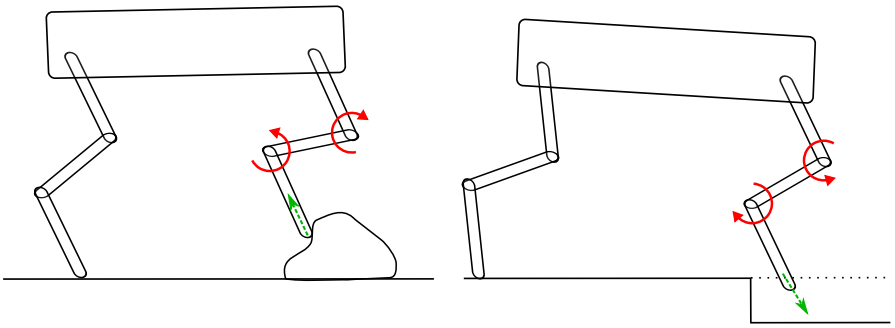


Figure 5.12: Reflexes. Left) Leg hits an obstacle in the swing phase. A stumbling correction reflex for extra knee flexion is activated. Right) A missing contact situation. Knee extension reflex increases the leg length to quickly acquire ground contact. Note that the legs follow a four-bar pantograph mechanism, so movements of knee and ankle joints are coupled.

5.2.3 Lateral Stepping Reflex

Examples of BigDog, HyQ, StarlETH, and other robots show how a lateral stepping is crucial to prevent falling after a large lateral perturbation. In a more general context, the capture point idea [Pratt et al., 2006] demonstrates how the stepping location can be calculated to capture the body from falling by taking a defined number of steps.

Most of the stepping strategies need a good estimation of the robot velocity. Estimating the robot's velocity is not an easy task, and needs proper state estimation, as shown for example in [Bloesch et al., 2013]. Moreover, assumptions like absence of slippage are taken into account.

For the case of our robots, state estimation is even more difficult, as they slip while locomoting, there are substantial bendings in the body parts, not all the joints are equipped with encoders, only binary contact sensing is available, and the processing power is quite limited. Here we use a simple approach to activate a lateral stepping reflex based on thresholding the lateral acceleration of the main body. Assume that the acceleration of the main body, sensed using an accelerometer, is $\mathbf{a}$, and $\mathbf{R}_{str}$ is trunk's straight rotation matrix (excluding yaw). We can calculate the lateral acceleration with respect to the world coordinates as:

$$\begin{bmatrix} \text{acc}_x \\ \text{acc}_y \\ \text{acc}_z \end{bmatrix} = \mathbf{R}_{str}\,\mathbf{a} \tag{5.31}$$

We record the values of acc_z (lateral acceleration w.r.t. the world) for the robot walking on flat ground without any perturbation, and for when laterally pushed. By comparing the two, we find a *threshold* value for the activation of the Lateral Stepping Reflex (LSR). After that, LSR is

defined as:

$$\begin{cases} a_{l,AA} \leftarrow k_{lsr} \frac{\text{acc}_z}{threshold} & \text{if acc}_z > threshold \text{ and } a_{l,AA} \approx 0 \\ \dot{a}_{l,AA} = -\alpha_{lsr} a_{l,AA} & \text{else} \end{cases} \quad l = 1..4 \tag{5.32}$$

where k_{lsr} is the activation gain. α_{lsr} is the reflex decay, defined such that it takes about two stride cycles before the reflex dies out. LSR is only activated until all other LSR have died out. $a_{l,AA}$ is the amplitude of the limit cycle for hip AA, and hip AA limit cycle is defined as:

$$f_{l,AA} = a_{l,AA} \sin(\theta_{l,AA}) \quad l = 1..4 \tag{5.33}$$

Since LSR is activated by amplifying the limit cycle of hip AA, and hip AA and hip PR are in phase ($\forall l : \phi_{l,AA,PR} = 0$), each leg naturally moves along the direction of the push while in the swing phase, and moves the opposite way in the stance phase to help the body move along the acceleration direction. At the same time, $a_{l,AA}$ decays, so smaller steps are taken as time passes by, until the perturbation is damped.

5.3 Summary

This chapter was dedicated to the design of posture control and reflex mechanisms for quadrupedal locomotion. While a CPG module, consisting of coupled morphed oscillators (Chapter 4), can be sufficient for flat terrain locomotion, sensory feedback should be included for the case of rough terrain locomotion. Posture control is needed to continuously adjust the robot's attitude and keep the body upright while locomoting on rough terrain. On top of the posture control module, reflexes were defined to handle recovery cases, where a quick and momentary reaction is needed.

We explained how kinematic model-based posture control and simple reflex mechanisms can be designed and incorporated into the CPG module performing the nominal pattern generation. We introduced three main ways to add feedback to the CPG module (a posture controller should use (1.) or (2.), but (3.) can be combined with both):

1. by adding adjustment torques to the feedforward torques generated by the CPG module;
2. through additive terms $\xi_{r,i}$ on the CPG output dynamics;
3. by modifying the limit cycle shapes $f_i(.)$ for each joint's CPG node.

The first way was implemented borrowing concepts from Virtual Model Control (VMC) to convert virtual forces to adjustment torques using the Jacobian transpose method. The virtual forces were defined such that they would keep the body upright. This approach needs the robot to have torque-control capability, and is only tested in simulation later in Chapter 7.

Since we assume that there is no torque-control capability due to the limitation of our small robots, we need to modify the first way. The second way was introduced by assuming that the virtual forces are virtual velocities, and then using the Jacobian inverse method to convert them to joint velocities. Since the CPG is generating joint velocities (before numerical integration), adding virtual velocity-based feedback becomes trivial through introducing additive terms. The second way was also used to implement stumbling correction and leg extension reflexes which are defined as decaying impulse signals to the CPG dynamics. Reflexes can also be combined with VMC posture controller, and in that case we simply set $\boldsymbol{\xi}_{pos} = \mathbf{0}$.

We used the third way to implement the angle-of-attack control and the lateral stepping reflex. Referring to Figure 4.12, when the limit cycle of a morphed oscillator is instantly changed, the oscillator states smoothly converge from the old limit cycle to the new one. We exploited this feature to modify the limit cycle shapes according to a change in ground inclination, or in response to a significant change in the lateral acceleration, while keeping the control input to the low-level controller continuous.

We will see in the next Chapter how the introduced modules are put together in a control architecture, along with additional modules to control the interaction between them. After that, Chapters 7 and 8 will demonstrate the results obtained by applying the proposed control architecture to simulated and hardware robots.

6 Control Architecture

Great things are done by a series of small things brought together.

Vincent Van Gogh

Here in this chapter we explain how the control modules, including the ones described previously, are put together. We aim to design a modular architecture which is capable of handling different locomotion control tasks like speed control, turning, reflex timings, etc. We explain what each module is, what are its inputs and outputs, and how it is set up and tuned. Figure 6.1 depicts the sketch of the proposed modular architecture.

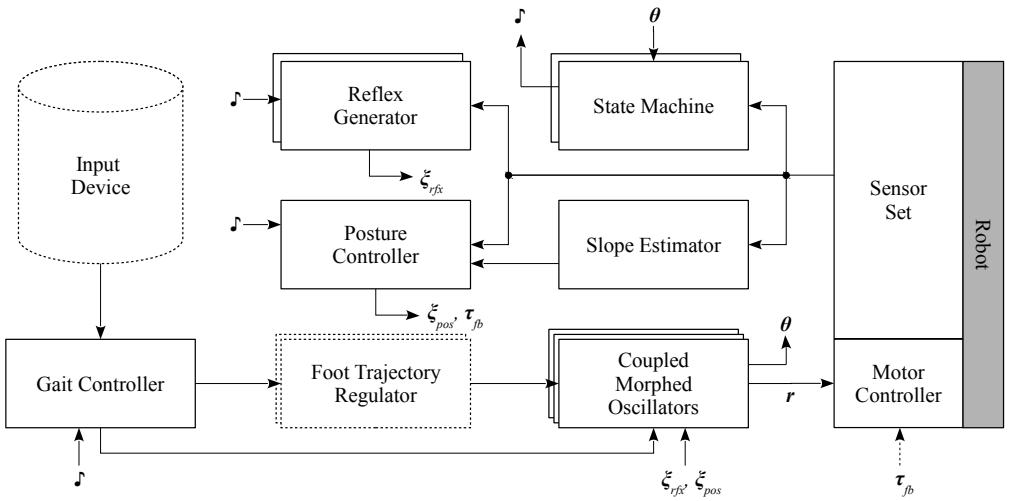


Figure 6.1: The proposed modular architecture. The dashed boxes are optional. The boxes that are repeated two (three) times are for when there is one module per leg (joint). Coupled morphed oscillators (CPG), posture controller, and reflex generator modules are the encapsulation of the ideas previously discussed in Chapters 4 and 5. Gait controller is responsible for regulating the speed and the turning rate. Foot trajectory regulator stores the desired foot trajectory, and modifies it as requested by the gait controller. Slope estimator calculates the local ground inclination. State machine controls the timing and activation of the control modules (♪). Note that $\boldsymbol{\tau}_{fb}$ is only if torque-control is available (in that case $\boldsymbol{\xi}_{pos} = \mathbf{0}$).

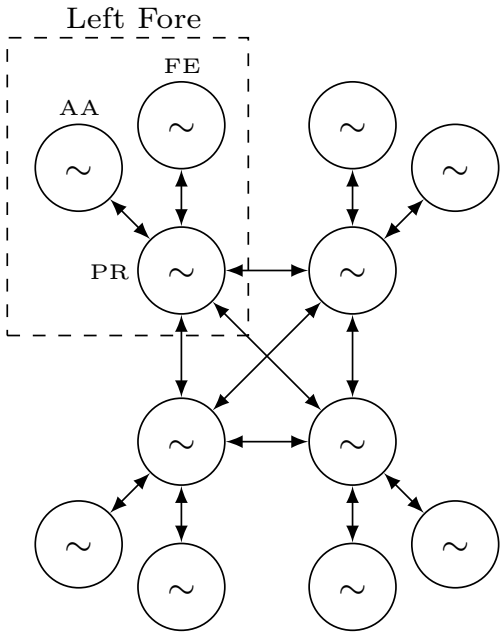


Figure 6.2: The coupling graph used in this thesis. Each leg (e.g. left fore) consists of three oscillators for hip AA, hip PR, and knee FE. Hip PR joints are the coupling link between different legs. All the couplings are bidirectional, and all the coupling weights are chosen to be the same.

6.1 Morphed Oscillators (MO)

Inputs: frequency ϑ (GC), limit cycle shapes $\mathbf{f}(.)$ (FT), phase lag matrix $\mathbf{\Phi}$ (GC), posture control feedback $\boldsymbol{\xi}_{pos}$ (PC), reflex feedback $\boldsymbol{\xi}_{rfx}$ (RG)

Parameters: coupling strength c, convergence rate γ

Outputs: motor commands $\mathbf{r}$ (MC), phases $\boldsymbol{\theta}$ (SM)

Description: Chapter 4 explained how nonlinear oscillators with desired limit cycle shapes can be designed using the concept of Morphed Oscillators. Here morphed oscillators are used to take the role of CPGs in locomotion. One first-order morphed oscillator is used per active degree of freedom and encodes the desired joint angle profile over stride phase. For this thesis we mainly use compensated unit-radius amplitude-controlled morphed oscillators:

$$\dot{\theta}_i = \Omega_i \tag{6.1}$$

$$\dot{r}_i = \Omega_i f_i'(\theta_i) + \gamma\left(f_i(\theta_i) - r_i\right) + \xi_{rfx,i} + \xi_{pos,i} \tag{6.2}$$

$$\Omega_i = 2\pi\vartheta + \sum_{j=1}^{N} c_{ij}\sin(\theta_j - \theta_i - \phi_{ij}) \tag{6.3}$$

where θ_i, Ω_i and r_i respectively are the phase, the coupling dynamics, and the output of the ith oscillator. γ is the convergence rate, ω is the locomotion frequency multiplied by 2π, and c_{ij} and ϕ_{ij} are the coupling strength and

phase difference between the ith and jth oscillators. $f_i(.)$ defines the shape of the limit cycle of the ith oscillator and $f_i'(\theta_i) = \partial f_i(\theta_i)/\partial\theta_i$. r_i is the joint angle reference for the ith DOF, and $\xi_{rfx,i}$ and $\xi_{pos,i}$ are reflex and posture control angular velocity feedbacks.

Tuning: The coupling parameters c_{ij} are set either to zero or a fixed value of $c = 5$, and are used to implement a desired coupling graph. For this thesis, we always use a coupling graph as shown in Figure 6.2. The convergence rate γ is set such that typical perturbations are damped in less than 10% of the stride duration. This varies depending on the stride duration, but based on a typical 0.400[s] stride duration, we fix the value to $\gamma = 50$ [1/s].

6.2 Posture Controller (PC)

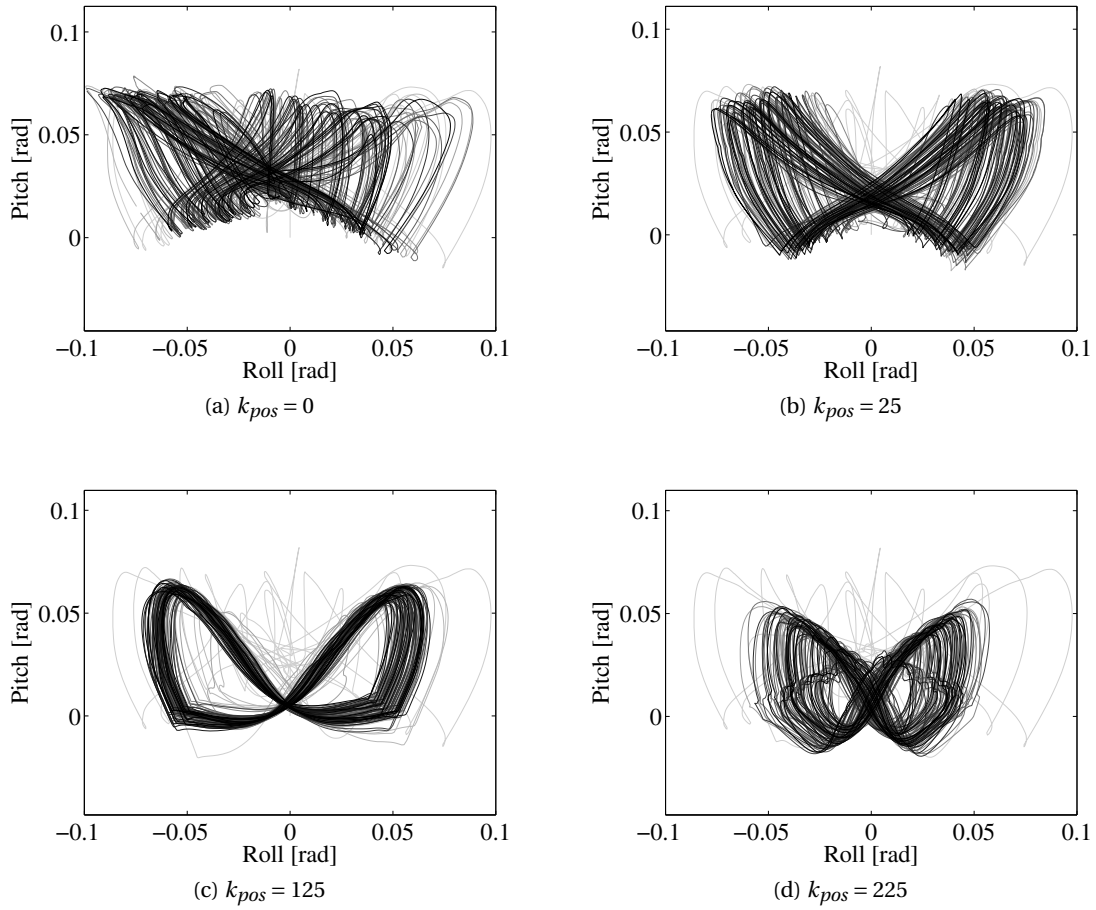
Inputs: estimated ground pitch $\psi_{pitch,gnd}$ (SE), sensed joint angles $\mathbf{q}$ (SS), robot rotation matrix $\mathbf{R}$ (SS), activation permission (SM)

Parameters: k_{pos} (or k_{att}, k_{lat} and k_{trn}), k_s

Outputs: posture control feedback $\boldsymbol{\xi}_{pos}$ (MO) (or $\boldsymbol{\tau}_{fb}$ (MC))

Description: As discussed in Chapter 5, posture control feedback signals can be generated as angular velocity feedback signals $\boldsymbol{\xi}_{pos}$ or torque feedback signals $\boldsymbol{\tau}_{fb}$. The posture control module encapsulates all the functionalities described in Chapter 5, Section 5.1. It receives the estimated ground pitch from the slope estimator module, and calculates the posture control feedback signals, taking into account the feedback gains. One (PC) is used for the whole robot.

Tuning: The only parameters which need to be tuned for this module are the feedback gains. For both the position- and torque-control modes, the feedback gains are tuned in a similar fashion. Let us assume that the robot is already walking on the flat ground with a provided (MO) module. We take the Role-Pitch-Variations (RPV) plot as a measure of how the posture control module is affecting the body attitude. RPV plot illustrates the relation between the roll and pitch of the robot's body, sampled through different phases of locomotion. This is similar to having numerous Poincaré sections [Teschl, 2012] based on the phase value, and put them together in one plot. What is important to look for in a RPV plot is the periodicity over different cycles, drift of RPV over time (or any quasi-periodic behavior), and the amplitudes of RPV. Figure 6.3 provides a sample of the RPV plot for the case of the Oncilla simulated robot. More data is provided later in Chapter 7. For the sake of the tuning process, the feedback gain k_{pos} is initiated with a zero value, and monotonically increased until the RPV becomes nearly optimal. By nearly optimal we mean having an improved periodicity and no drift. Excess increase of the feedback gain k_{pos} results in degeneration of the RPV plot, as shown in Figure 6.3. For torque



(a) $k_{pos} = 0$ (b) $k_{pos} = 25$

(c) $k_{pos} = 125$ (d) $k_{pos} = 225$

Figure 6.3: RPV for different posture control gains. A good value of k_{pos} stabilizes the RPV and makes it more periodic (c), however excessive increase can have a counter effect (d).

control case, we set k_{att} the same way we set k_{pos}, and $k_{lat} = k_{trn} = \frac{1}{10} k_{att}$ to give relative importance to attitude control.

We then set the angle-of-attack gain k_s. For the hip offset strategy, k_s is set such that the robot could go down a 20% slope. We start with a default value of 1, and then slightly increase it to obtain the desired performance. This gives a value of $k_s = 1.25$. Similarly for the foot trajectory case, the foot trajectory shifting gain k_s is set such that the robot could go down a 20% slope. This yields a value of $k_s = 0.2$.

6.3 Reflex Generator (RG)

Inputs: lateral acceleration acc_z, activation permission (SM)

Parameters: reflex gain k_{rfx}, sidestep activation k_{lsr}

Outputs: reflex feedback $\boldsymbol{\xi}_{rfx}$, sidestep amplitude $a_{l,AA}$

Description: Chapter 5 explained why reflexes are needed in order to locomote on rough terrain and quickly react to unwanted events. (RG) module encapsulates the three reflexes described earlier: the stumbling correction reflex (SCR), the leg extension reflex (LER), and the lateral stepping reflex (SSR). This module generates the angular velocity reflex signals and the AA amplitudes sent to (MO). One (RG) is used per leg.

Tuning: k_{rfx} is set incrementally until the robot can go over a vertical obstacle, or a downward step, placed on a flat terrain. The obstacle height is set to 20% of the leg length. This gives $k_{rfx} = 150$ for the simulated torque-controlled quadruped, and $k_{rfx} = 50$ for simulated Oncilla. For the hardware Oncilla robot, smaller obstacles of about 10% of the leg length are used and $k_{rfx} = 200$ (hardware robot's low-level controller is slower and needs a bigger gain, as we will see later in Chapter 8).

For the LSR, we experiment with lateral external forces while LSR is off, and increase the force magnitude until the robot becomes unable to handle them. Afterwards, we increase k_{lsr} such that the robot initiates a lateral stepping sequence given a similar force magnitude. This gives a typical value of $k_{lsr} = 0.1$ with a lateral acceleration *threshold* of $1.5g[\mathrm{m/s^2}]$.

6.4 State Machine (SM)

Inputs: phases $\boldsymbol{\theta}$ (MO), output derivatives $\dot{\mathbf{r}}$ (MO), contact state $\boldsymbol{\delta}$ (SS)

Parameters: grace phase difference $\Delta\theta_{grace}$, swing phase span $\Theta_{l,sw}$, stance phase span $\Theta_{l,st}$

Outputs: activation permission for (PC) and (RG), update permission for (GC)

Description: State Machine is one of the key modules which has not been described before. The role of (SM) is to determine the timing and activation of the other modules. To construct (SM), we first define simple rules on when each module should be activated. The rules are based on the phase of each leg's $\theta_{l,PR}$ and contact state of the leg δ_l, and the current state of (SM). Figure 6.4 depicts the state machine, with a full description in the caption. One separate (SM) is used per leg. Posture control feedbacks for the lth leg are taken into account only when the (SM) for that leg is in the *Stance* state. For reflexes (RG), SCR is active if the leg acquires contact in the swing while protracting, and LER gets activated if there is a missing contact. LSR is active all the time. Lastly, gait controller (GC) is only updated once per cycle (for each leg) when the phase passes the mid-swing (not depicted).

Tuning: The important parameters to tune for (SM) are the swing and stance phase spans $\Theta_{l,sw}$ and $\Theta_{l,st}$. These phase spans are simply obtained by extracting the swing and stance onsets from the robot running on flat ground. In theory $\Theta_{l,sw}$

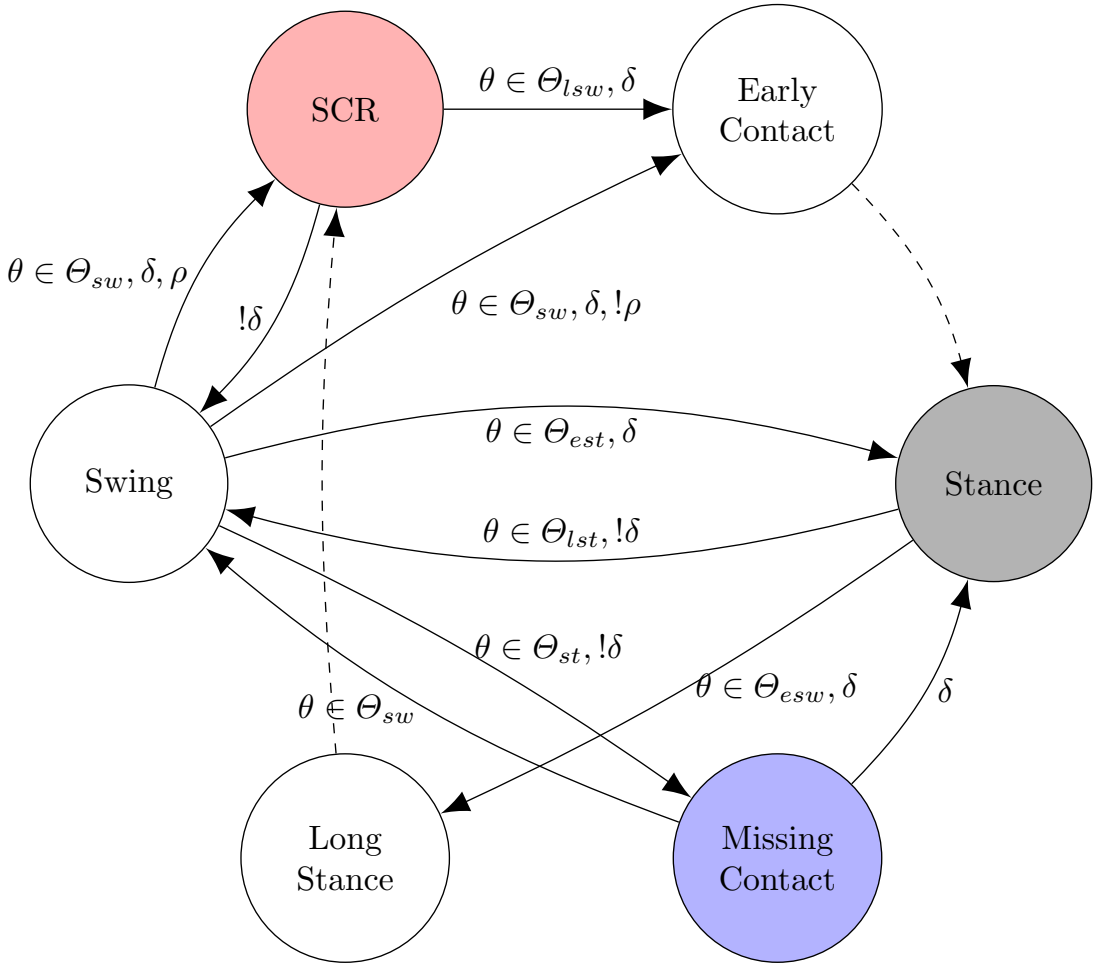


Figure 6.4: The state machine (for one leg). In a perfect situation on flat terrain, the robot only switches between *Swing* and *Stance* states. However locomotion on flat terrain can include *Early Contact*s, *Missing Contact*s, and unexpectedly *Long Stance* phases. The state machine uses the on/off contact status $\delta = \delta_l$ and the hip PR phase $\theta = \theta_{l,PR}$ as inputs. The stumbling correction reflex (SCR) is activated in the *SCR* state (when a swing leg hits an obstacle), and the leg extension reflex (LER) is activated in the *Missing Contact* state. For each leg, the posture controller feedback is taken into account only when the leg is in the *Stance* phase. For this figure, Θ_* is the phase span each state is active in, extracted from a flat terrain run. $*$ can be early swing (esw), swing (sw), late swing (lsw), early stance (est), stance (st), late stance (lst). Being late or early is determined by the value of $\Delta\theta_{grace}$. ρ determines whether a leg is protracting, which is 1 if $\dot{r}_{l,PR} > 0$. The dashed arrows indicate immediate transitions. The immediate transition from *Long Stance* to *SCR* is only applied in simulation, and *Long Stance* transits to *Swing* in the hardware robot. This is based on personal experience, and due to large variations of the stance duration when the hardware robot is not calibrated precisely.

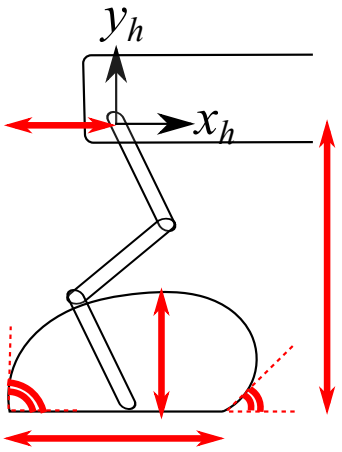


Figure 6.5: Foot trajectories are defined in the Sagittal plane, w.r.t. the shoulder/hip frame (x_h, y_h), and are defined by the step length, the foot clearance, the touchdown angle, and the liftoff angle.

and $\Theta_{l,st}$ for all the legs should be the same if the robot is performing a perfect trot gait, but typically, due to the robot's mass distribution, they are different for front and hind legs. Left-right symmetry is however well respected.

The other parameter to tune is the grace phase percentage $\Delta\theta_{grace}$, which prevents a reflex activation if there is only a slight time mismatch. For example, LER will only get activated if the missing contact state is extended by $\Delta\theta_{grace}$. We arbitrarily set $\Delta\theta_{grace}$ to 5% of the cycle time.

6.5 Foot Trajectory Regulator (FT)

Inputs: swing time t_{sw} (GC), duty factor d (GC), desired velocity v_f (GC), asymmetric step length magnification $a_{l,asym}$ (GC)

Parameters: foot trajectory location, foot clearance, touchdown angle, liftoff angle

Outputs: joint-space limit cycles $\mathbf{f}(.)$ (MO)

Description: Foot trajectory regulator is an optional module if the reference joint angle profiles are being calculated from foot trajectories. Foot trajectories are defined as shown in Figure 6.5, and are parametrized by the location w.r.t. the hip coordinates, step length, the foot clearance, the touchdown angle, and the liftoff angle. One (FT) is used per leg. Based on the constant swing time t_{sw}, the desired locomotion speed v_f and the given duty factor d, the step length can be calculated as:

$$\Delta x_{l,step} = a_{l,asym} \frac{d}{1-d} t_{sw} v_f \tag{6.4}$$

Swing and stance phase spans are also calculated with respect to d. Assuming that the swing is defined by $\theta \in [0, \pi)$ and stance by $\theta \in [\pi, 2\pi)$ for $d = 0.5$

(transition phase $\theta_{transition} = \pi$), the transition phase for an arbitrary d is:

$$\theta_{transition} = 2(1-d)\pi \tag{6.5}$$

Knowing the step length, the desired phase spans for swing and stance, and the other parameters defining the foot trajectory, we use closed-form planar inverse kinematics of the leg to calculate the joint angle limit cycles $\mathbf{f}(.)$. Please note that the closed-form inverse kinematics does not take into account the compliance of the legs, and performing the calculated joint angle profiles does not lead to a perfect recreation of the foot trajectory.

Tuning: Other than the step length, other parameters of the foot trajectory are manually tuned. The location of the foot trajectory is set to be below the shoulder/hip axis, with a distance of about 0.75% of the leg length. Foot clearance is set to 25% of the leg length in simulation, and about half of that for the hardware robot. Touchdown angle is set to about $\frac{\pi}{4}$ to initiate leg retraction before the onset of the stance phase, and liftoff angle is typically set to $\frac{\pi}{2}$ to have a clear liftoff, or about $2\frac{\pi}{3}$ to have a follow-through.

It is worth mentioning that it can be an interesting scientific question to explore how the different parameters defining the foot trajectory affect the dynamics of locomotion including balance, energy efficiency, actual duty factor (which can be different from the commanded one), etc.

6.6 Gait Controller (GC)

Inputs: desired velocity v_f (ID), desired turning rate ϖ (ID), update permission (SM)

Parameters: swing time t_{sw}, phase lag matrix and duty factor for different gaits $\mathbf{\Phi}_{gait}$, d_{gait}

Outputs: desired velocity v_f (FT), duty factor d (FT), asymmetric step length magnification $a_{l,asym}$ (FT), phase lag matrix $\mathbf{\Phi}$ (MO), locomotion frequency ϑ (MO)

Description: Gait controller is responsible for adjusting the gait parameters based on requests for speed or direction change. One (GC) is used for the whole robot. Each gait type (e.g. walk, trot) have a specific span of validation defined by trapezoid membership functions. The lateral sequence walk is active for speeds in the range $[0, 0.2]$[m/s] and the trot is active for speeds above that. There is a transition span between the two gaits, which has been set to $0.05[m/s]$. Each gait is defined by a specific phase lag matrix $\mathbf{\Phi}_{gait}$, and a specific duty factor d_{gait}. The output phase lag matrix fed to the (MO) is calculated as:

$$\mathbf{\Phi} = \sum_{i}^{\text{All gaits}} \beta_i(v_f)\mathbf{\Phi}_i \tag{6.6}$$

where $\beta_i(v_{forward})$ is the activation value of the ith gait based on its membership function, and v_f is the desired forward locomotion speed. This provides

a soft switching of the phase lag matrix, similar to [Coros et al., 2011]. The effective duty factor is also calculated in a similar fashion, resulting in d.

The stepping frequency is defined by the swing time (which is constant for different v_f) and d:

$$\vartheta = \frac{1-d}{(1+d)\,t_{sw}} \tag{6.7}$$

□

Gait controller utilizes three different ways to implement the turning behavior:

1. Generating turning posture control commands for (PC);
2. Modifying the hip AA limit cycles for (MO);
3. Asking for step length modification from (FT).

The first method is described in Chapter 5, and the role of (GC) is only to provide the magnitude of the desired turning virtual forces (torque-control case) or the yaw rotation offset (position-control case), proportional to the desired turning rate ϖ.

For the second method, as explained before, a zero-amplitude sine-wave is embedded into the (MO) nodes for the AA joints (Chapter 5, Section 5.2.3). What (GC) does is setting the amplitude of the fore and hind hip AA joints with opposite signs, proportional to the desired turning rate:

$$f_{l,AA} = a_{l,AA}\sin(\theta_{l,AA}) \quad l = 1..4 \tag{6.8}$$

$$a_{l,AA} = \lambda \Delta\psi_{yaw,des} \tag{6.9}$$

$$\lambda = \begin{cases} +1 & l \in \{LF, RF\} \\ -1 & l \in \{LH, RH\} \end{cases} \tag{6.10}$$

Please note that $a_{l,AA}$ is only set if there is no LSR activated, since $a_{l,AA}$ is also used for LSR which has a higher importance.

For the third method, step length can be modified in order to implement turning without the use of the AA joints. We implement a turning strategy used typically for two-wheeled mobile robots. We decrease the step length for the legs on one side of the body to turn in that direction. The robot will turn in-place if the step lengths are the same but with opposing signs. We have:

$$a_{l,asym} = \begin{cases} 2\varpi + 1 & \varpi < 0 \;\&\; l \in \{LF, LH\} \\ 1 & \varpi > 0 \;\&\; l \in \{LF, LH\} \\ 1 & \varpi < 0 \;\&\; l \in \{RF, RH\} \\ 1 - 2\varpi & \varpi > 0 \;\&\; l \in \{RF, RH\} \end{cases} \tag{6.11}$$

where $a_{l,asym}$ is the step length amplifier, later given to (FT). This approach produces a small amount of slippage because the contact legs (diagonal pairs

for the trot gait) are not on the same axis (as opposed to a two-wheeled mobile robot, like [Mondada et al., 2009]).

The first turning method is only used in simulation where the sensing of trunk yaw is accurate. The second method is used on the hardware robot for most of the turning cases, however fast turning with this method results in considerably large forces on the hip AA axis. The third method is mostly used to turn when the ground is slippery or on rough terrain, as it does not depend on lateral movement of the feet which can cause stumbling to an obstacle on rough terrain (SCR is not implemented in the transverse plane).

Tuning: Swing time t_{sw} is the key parameter of (GC). House cats, which are slightly bigger than our robots in terms of size and weight, use a rather constant swing time of about 0.3[s] [Halbertsma, 1983]. As the stride duration decreases having a shorter leg length [Alexander, 1984], we use a range of $[0.15, 0.25]$[s] for swing time. A shorter value for swing time is chosen if the robot is supposed to locomote faster.

The other parameters to set for (GC) are the phase lag matrix and duty factor for each gait. We define the phase lag matrix for lateral sequence walk and trot similar to [Alexander, 1984]. Walk duty factor is set to 0.7, and trot duty factor is set to 0.5 for simulation, and 0.55 for hardware.

6.7 Slope Estimator (SE)

Inputs: sensed joint angles $\mathbf{q}$ (SS), straight rotation matrix $\mathbf{R}_{str}$ (SS)

Parameters: zero threshold ψ_0

Outputs: estimated ground pitch $\psi_{pitch,gnd}$ (PC)

Description: We use the straight rotation matrix (from IMU) and the sensed joint angles (from encoders) to estimate the ground inclination on-the-fly as the robot walks on it (Figure 6.6). Assume that $\mathbf{p}_{f,r}$ is the coordinates of a fore foot in contact with the ground, and $\mathbf{p}_{h,r}$ a hind foot, both w.r.t. the frame attached to the robot (these are the diagonal pairs for a trotting robot). The coordinates of the feet positions w.r.t. the world (ignoring yaw) are:

$$\begin{bmatrix} \mathbf{p}_{f,w} & \mathbf{p}_{h,w} \end{bmatrix} = \mathbf{R}_{str} \begin{bmatrix} \mathbf{p}_{f,r} & \mathbf{p}_{h,r} \end{bmatrix} \tag{6.12}$$

and knowing them, and assuming $\Delta\mathbf{p} = \mathbf{p}_{f,w} - \mathbf{p}_{h,w}$, the ground pitch inclination is:

$$\psi_{pitch,gnd} = \tan^{-1} \frac{\Delta\mathbf{p}_{[y]}}{\Delta\mathbf{p}_{[x]}} \tag{6.13}$$

and we set $\psi_{pitch,gnd}$ to zero if it is smaller than a threshold value ψ_0, to ignore the estimation noise while walking on flat terrain.

It is important to mention that at least three contact points are required to

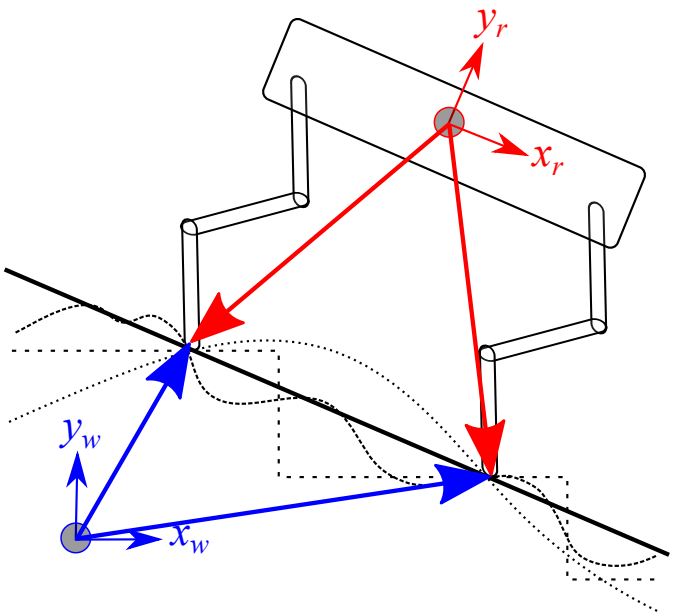


Figure 6.6: Slope estimation. First (drawn in red), the positions of the feet w.r.t. the robot frame are calculated using forward kinematics. Second (drawn in blue), the positions of the feet w.r.t. the world coordinates are calculated utilizing the rotation matrix. Finally, the projection of the feet positions in the Sagittal plane are used to calculated the ground pitch. Note that from a kinematic point of view, the estimated inclination is the same for different ground geometries (dashed and dotted lines) as long as the positions of contact points are the same. The only way to differentiate between different ground geometries is to utilize the ground reaction forces.

estimated ground roll and pitch at the same time. Since in trotting there are (mostly) two feet on the ground simultaneously (with no or very short periods of three/four contacts), we have to assume a known value for ground roll, which we assume to be zero. This can be solved by either having longer three/four contact periods, or by taking into account the ground reaction forces.

Moreover, the inclination estimation here does not tell about the local ground inclination under one feet, and instead calculated the inclination of the plane which passes through the contact feet. Hence, we cannot differentiate between different ground geometries if they have the same contact locations, as shown in Figure 6.6. Again, ground reaction forces can help to solve this issue, but we not address this problem in the context of this thesis, as we assume that only binary contact information is available.

Tuning: The only parameter to set for (FT) is the zero threshold ψ_0. Based on the typical measurement errors for slope estimation (on the hardware robot), ψ_0 is set to 0.05[rad].

6.8 Motor Controller (MC)

Inputs: joint position commands $\mathbf{r}$ (MO), adjustment torques $\boldsymbol{\tau}_{fb}$ (PC), sensed joint angles $\mathbf{q}$ (SS)

Parameters: proportional gain k_p, derivative gain k_d

Outputs: motor control voltages/torques

Description: The role of the motor controller is to convert position commands to motor control signals. The motor control signals are voltages (with current limitation) for the Oncilla robot (simulation and hardware), and torques for the simulated torque controlled quadruped. In case of the Oncilla robot, $\mathbf{r}$ is simply transferred to the low-level motor controller. For the case of the torque controlled quadruped, we use a PD controller:

$$\boldsymbol{\tau}_{out} = k_p(\mathbf{r} - \mathbf{q}) + k_d(\dot{\mathbf{r}} - \dot{\mathbf{q}}) + \boldsymbol{\tau}_{fb} \tag{6.14}$$

where $\boldsymbol{\tau}_{fb}$ is the vector of adjustment torques provided by (PC).

Tuning: For the case of the torque controlled simulated quadruped, we do not choose high values for k_p as it makes the robot very stiff, and k_p is set high enough to generate acceptable joint amplitudes. k_d is set to zero, as the the robot is physically damped for better simulation stability. For the Oncilla robot (simulation and hardware), k_p and k_d are set internally in the motor drivers.

6.9 Sensor Set (SS)

Inputs: Raw sensor data

Parameters: contact threshold

Outputs: sensed joint angles $\mathbf{q}$, on/off contact $\boldsymbol{\delta}$, rotation matrix $\mathbf{R}$, straight rotation matrix $\mathbf{R}_{str}$, lateral acceleration acc_z

Description: Sensor set encapsulates the sensor devices and reformats and prepares their data for the other modules. $\mathbf{q}$ is sensed using magnetic encoders, and is scaled and offset to be in the same coordinate system as of (PC). For the Oncilla hardware robot, hip AA is not equipped with encoders, and we use the last commanded joint angle as the sensed one.

On/off contact sensing in simulation is simply done by detecting if there is a collision between the feet bounding objects and other objects (including internal collisions). For the Oncilla hardware robot, we have tried four different approaches for contact sensing, as explained later in Chapter 8. For all the approaches, a continuous signal is read and thresholded to detect contact.

Rotation matrix and acceleration are directly read from the internal physics engine in simulation, and from a high-end IMU on the hardware. The straight rotation matrix is then calculated as explained in Chapter 5, Section 5.1.1, by

canceling the yaw. lateral acceleration is calculated by pre-multiplying the sensed acceleration vector by the rotation matrix.

Tuning: The only parameter to tune is the sensitivity of the contact thresholding, which is done by keeping the robot in the air, lightly tapping on the feet, and finding the threshold from the recorded data. For the latest senors used (Optoforce 3D force sensors), the on/off contact threshold is 1[N].

6.10 Input Device (ID)

Inputs: Keyboard, joystick, etc

Parameters: increment steps

Outputs: desired velocity v_f, desired turning rate ω, turning stratefy

Description: This module is an optional way to send user commands to (GC). The values of v_f and ω are zero initiated, and changed by fixed incrementation. We use the standard input-output API of the Linux operating system to read commands sent via keyboard. We use the Controller Mate software [con, 2014] to use other devices (e.g. PS3 controller) by binding them to keyboard keystrokes. Keyboard commands are sent to the robot via secure shell (SSH).

Tuning: The increment steps are 0.02[m/s] for v_f, 0.01[rad] for turning by a change of yaw offset, 0.01[rad] for turning by amplifying hip AA movement, and 0.05[] for turning by asymmetrically changing the step length. These values are set manually. Nevertheless, the control is not sensitive to the choice of these values. Especially for a change of the desired speed by a big step, what happens is that the foot trajectories are recalculated and (MO) limit cycles are updated, and the (MO) states smoothly converge to the new limit cycle based on the value of γ, a parameter of (MO).

6.11 Summary

In this chapter we explained how the we encapsulate the functionalities, including the ones in Chapters 4 and 5. To summarize, we defined the following modules:

Morphed Oscillators (MO) Coupled morphed oscillators, as defined in Chapter 4, responsible for generating the nominal pattern of locomotion. One morphed oscillator is defined for each active degree of freedom;

Posture Controller (PC) Responsible for keeping the body upright by generating angular velocity feedback signals (or adjustment torques). As explained in Chapter 5, the adjustment or feedback signals are generated by introducing virtual velocities (or forces), and applying the Jacobian inverse (or transpose) method;

Reflex Generator (RG) Responsible for reacting to unwanted events in a quick and momentary fashion, as explained in Chapter 5;

State Machine (SM) Controls the timing and activation of (PC) and (RG), and tells (GC) when it can apply the updates;

Foot Trajectory Regulator (FT) Modifies the step length as needed, and recalculates the joint angle limit cycles for (MO);

Gait Controller (GC) Responsible for modifying the step length based on the desired speed and turning rate, and calculating the effective phase lag matrix and duty factor;

Slope Estimator (SE) Estimates the ground pitch inclination on-the-fly as the robot locomotes on different surfaces;

Motor Controller (MC) Performs low-level motor control, mostly using PID control;

Sensor Set (SS) Encapsulates sensor devices and reformats their data;

Input Device (ID) Binds different input devices to the standard input and transfers user commands to (GC) via SSH.

The modules defined here are not necessarily novel, and the contribution mainly lies in (MO), (PC), and partially in (RG) and (GC). We will see the results of applying the proposed control architecture on simulated and hardware robots in the next part of this thesis.

Experiments Part III

7 Simulations

For the things we have to learn before we can do them, we learn by doing them.

Aristotle

Publication

Parts of the material presented in this chapter is taken from:

- Mostafa Ajallooeian, Soha Pouya, Alexander Sproewitz, and Auke J Ijspeert. Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3321–3328. IEEE, 2013.
- Mostafa Ajallooeian, Sebastien Gay, Alexandre Tuleu, Alexander Sprowitz, and Auke J Ijspeert. Modular control of limit cycle locomotion over unperceived rough terrain. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3390–3397. IEEE, 2013.

For the second paper, all the credit for the simulator design goes to Alexandre Tuleu.

This chapter presents the result of applying the control architecture described in the previous chapter to simulated quadrupedal robots. Two different robots are used for experimentation, a stiff-by-nature torque-controlled (Ghostcat), and a passively-compliant position-controlled quadruped (Oncilla). Ghostcat is used to implement our ideas when we have assumed that torque-control is available (VMC posture control). Oncilla is then used to test the control architecture with a position-controlled robot, and prepare the control to be ported to the hardware robot. We use different benchmark tests including locomotion on uneven terrains and slopes, handling external perturbations, carrying asymmetric load, etc to evaluate the performance. For each simulated platform, the control parameters are fixed for all the scenarios, as a notion of generalization. The hardware results are later presented in Chapter 8.

Table 7.1: Body and control parameters of the simulated quadruped Ghostcat

Property	Value
Total mass	5.75[kg]
Head to total mass percentage	7%
Limb to total mass percentage	7%
Headless length	0.4[m]
Sagittal shoulder to hip distance	0.3[m]
Lateral hip to hip distance	0.24[m]
Standing leg length	0.28[m]
Limb segment length	0.14[m]
Foot radius	0.035[m]
Foot width	0.025[m]

Control parameter	Value
k_{att}	250
k_{lat}, k_{trn}	25
k_{rfx}	150

7.1 Ghostcat: Torque-controlled Quadruped

To test out initial ideas on using VMC to generate posture control feedback, with the assumption that torque-control is possible, we have made a torque-controlled quadruped called Ghostcat. Ghostcat is stiff-by-nature and does not include any passive compliance in the joints. It is modeled in Webots robot simulation software [Michel, 1998, 2004], which uses Open Dynamics Engine (ODE) [Smith, 2003] as the physics engine. The quadruped is about the weight and size of a cat (Table 7.1). All the body parts have uniform density distribution. There are three active DOF per leg, first hip abduction/adduction (AA, lateral hip joint), then hip protraction/retraction (PR, Sagittal hip joint), and finally knee flexion/extension (FE, Sagittal knee joint). All the joints are passively damped to increase the numerical stability of the simulation (with a damping factor of $b = 1$). There is no displacement between hip AA and hip PR joints. All limb segments have equal lengths. The robot is equipped with four on/off contact sensors (bumpers) around the feet, encoders for joint angle sensing, and an absolute rotation sensor placed in the trunk.

7.1.1 Control Parameters

In Chapter 6 we described how the control parameters are tuned. The value of control gains are given in Table 7.1. The only modification here is that we have not used a foot trajectory regulator (FT) for the experiments with Ghostcat. Instead the limit cycle profiles of the morphed oscillator nodes are defined as piecewise cubic Hermite polynomials [Fritsch and Carlson, 1980] with 4 equally spaced knots in $\theta \in [0, 2\pi)$, for each joint. This gives 16 parameters that can be set to tune the open-loop gait generated by CPG. One can use optimization

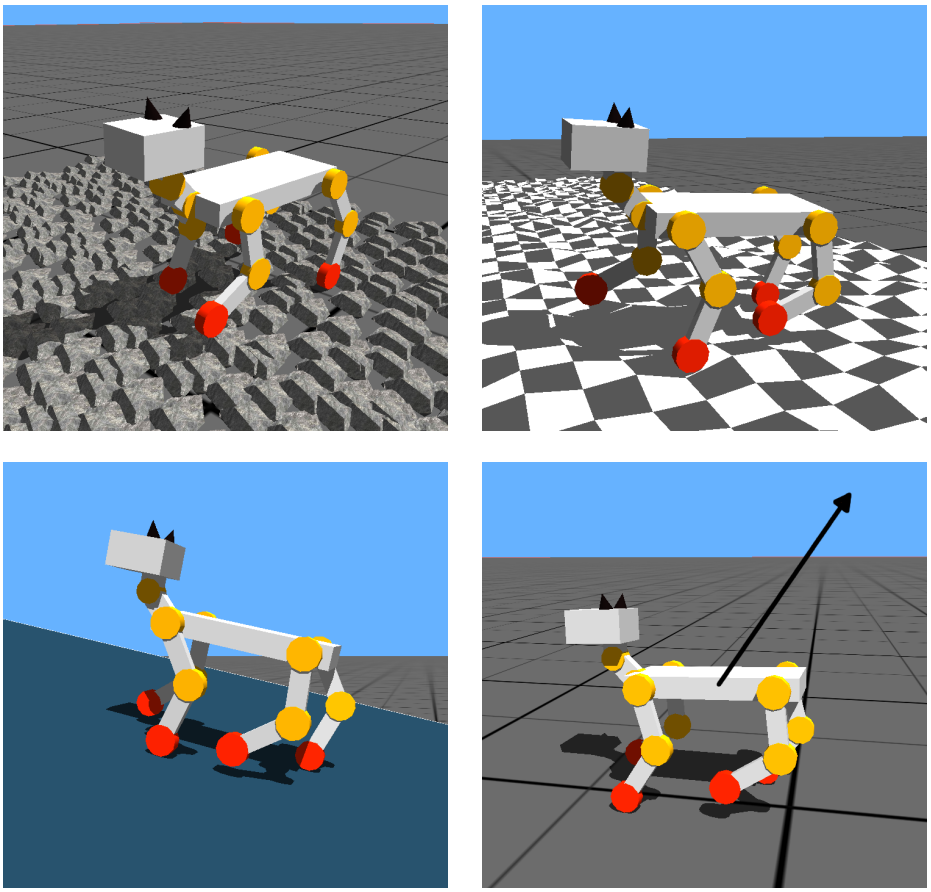


Figure 7.1: Ghostcat in different rough terrain scenarios. *Top-left,* passing through a rocky setup. *Top-right,* traversing over randomized uneven terrain. *Bottom-left,* climbing over slopes. *Bottom-right,* Handling external perturbations.

techniques to find a proper CPG gait [Tuleu et al., 2011], however, we did not find that necessary, and it took about 25 manual trials to find an acceptable gait. We start by front-hind symmetry and sinusoidal joint angle profiles and then slightly alter them. We also benefit from the ideas given in [Spröwitz et al., 2013] to use single-peak hip PR and double-peak knee FE profiles. The desired limit cycle for all AA joints are set to zero (vertical standing posture).

We implement and experiment with trot gaits, so the phase difference for ipsilateral and contralateral pairs are similarly π[rad], and 0[rad] for the diagonal pairs. A phase difference of $\frac{\pi}{4}$[rad] is introduced between hips and knees. These values are always fixed and we do not use them to tune the controller.

It is important to mention that we have omitted angle-of-attack control and slope estimation for the experiments with Ghostcat (they were not yet implemented at the time of experimenting). So the robot tried to keep the body orthogonal to the gravity vector.

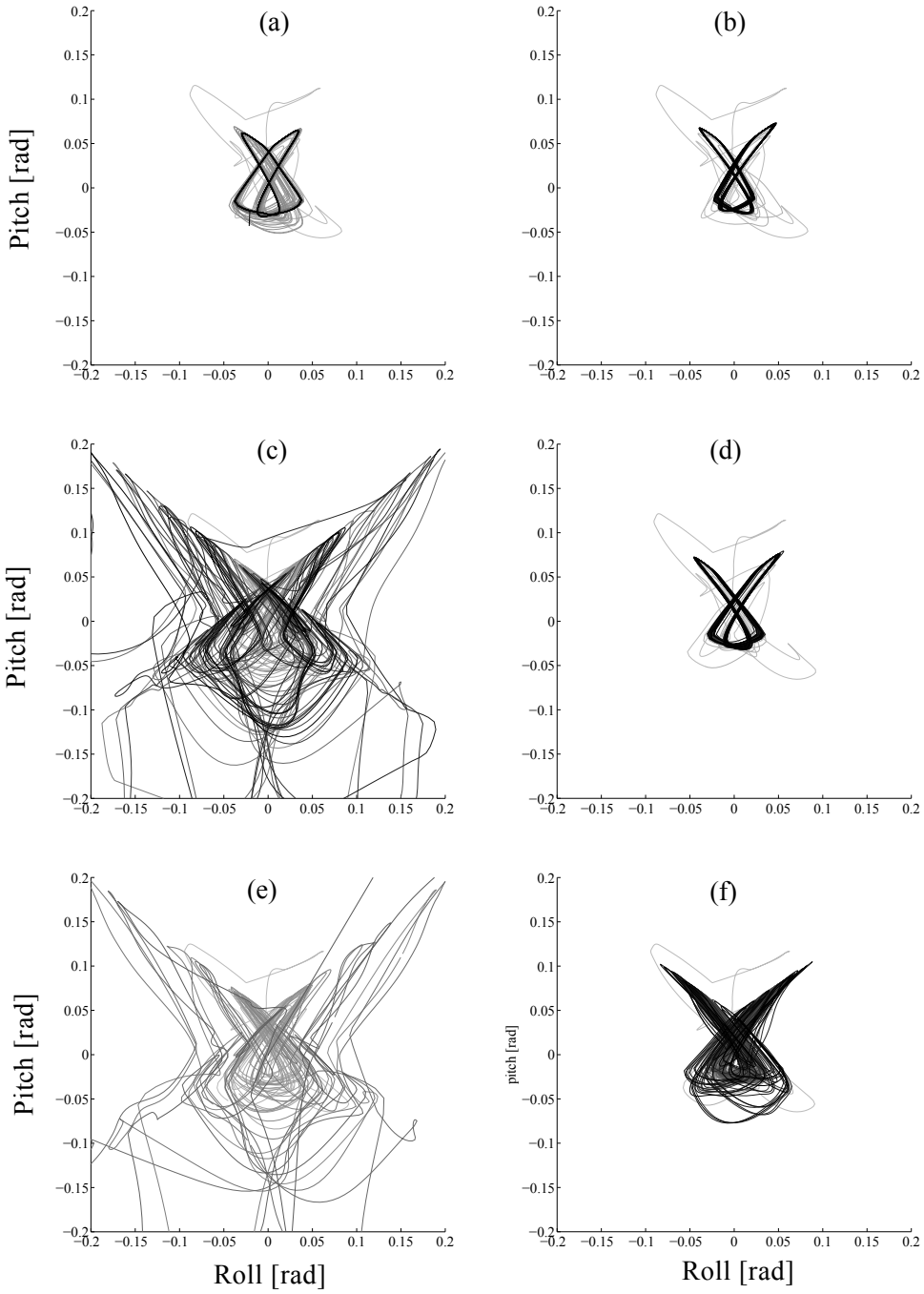


Figure 7.2: Comparison between the roll-pitch variations (RPV) of open-loop gaits (left) and their closed-loop counterparts (right). All the plots are for 40[s] runs equal to 80 gait cycles. The line color starts from gray in the beginning of the recording and becomes black in the end of recording at $t = 40$[s]. (a) A not-well-tuned open-loop gait. (b) Same gait as 'a', but reflex and posture controller modules are active. (c) A bad open-loop gait. The gait behavior is not regular. (d) Same gait as 'c', but the loop is closed. The RPV is much more periodic. (e) A not-open-loop-stable gait. The robot falls after about 20[s]. (f) The not-open-loop-stable gait can be turned into a sufficiently stable gait by closing the loop. The RPV are bounded, but of course not as regular as 'b' and 'd' because of the open-loop gait used.

7.1.2 Experimental Setup

We test our simulated robot in the following scenarios:

- passing through a rocky setup;
- traversing over randomized uneven terrain of $7-11\%$ of leg length variation in $0.1[m]$ spaced vertices;
- going over $14-21\%$ slopes;
- handling external pushes of $\{15[N], 0.5[s]\}$ magnitude while locomotion over flat terrain.

The first three types of the aforementioned scenarios are repeated from 25 different initial conditions. The last scenario is repeated 27 times where an external force of 15[N] pushes the robot for a duration of 0.5[s] with a random timing and directed towards the corners, edge centers, and face centers of a cube around the robot. All the experiments are executed in a time window of 20[s] out of which the first $5-7$[s] are used for initiation on a flat terrain and unperturbed. Figure 7.1 illustrates these scenarios. In all of these scenarios the robot does not have *any* prior knowledge about the environment, even the subtle information of "what kind of environment am I facing?". So there is no possibility to anticipate. All the experiments are done with a trot gait at 2[Hz] ($t_{sw} = 0.25$[s], $d = 0.5$) which gives an average speed of more than 0.6[m/s] equal to 2[BL/s].

7.1.3 Results

Before presenting the results obtained from the rough terrain experiments, we would like to show the effect of the virtual model controller on badly designed open-loop gaits, and on increasing the periodicity of the gait. Figure 7.2-(a) shows the Roll-Pitch-Variations (RPV) of a typical not well-tuned open-loop gait on flat terrain, where the settling time is long (about 12[s]). Figure 7.2-(b) depicts the RPV of the mentioned gait, but now with the posture controller active, which shows a better periodicity and faster settling (about 4.5[s]). The RPV is illustrated for an ill-parametrized open-loop gait in Figure 7.2-(c). Flat terrain locomotion with this gait did not lead to a fall, but the robot was constantly perturbed, and as it is obvious, the RPV is not periodic. We get Figure 7.2-(d) by activating the posture controller for this gait. Again, the RPV is greatly shrunk, and the outcome looks more periodic and symmetrical. A similar experiment is presented in 7.2-(e-f), however this time the open-loop gait is not even open-loop stable and the robot falls after about 20[s] of locomotion. After activating the posture controller the robot could locomote for more than 100[s] where we stopped the recording. For all these experiments the same parameters given in Table 7.1 are used (other than $f_i(.)$ which are used to generate different open-loop gaits).

Figure 7.3 presents the results obtained for the rough terrain experiment scenarios. Though the open-loop gait is stable on the flat terrain, it is mostly unsuccessful in the rough terrain scenarios. In case of rocky and uneven environments, there are two typical fall cases: 1) stumbling, and 2) bad foothold leading to unwanted body roll and pitch which are not cor-

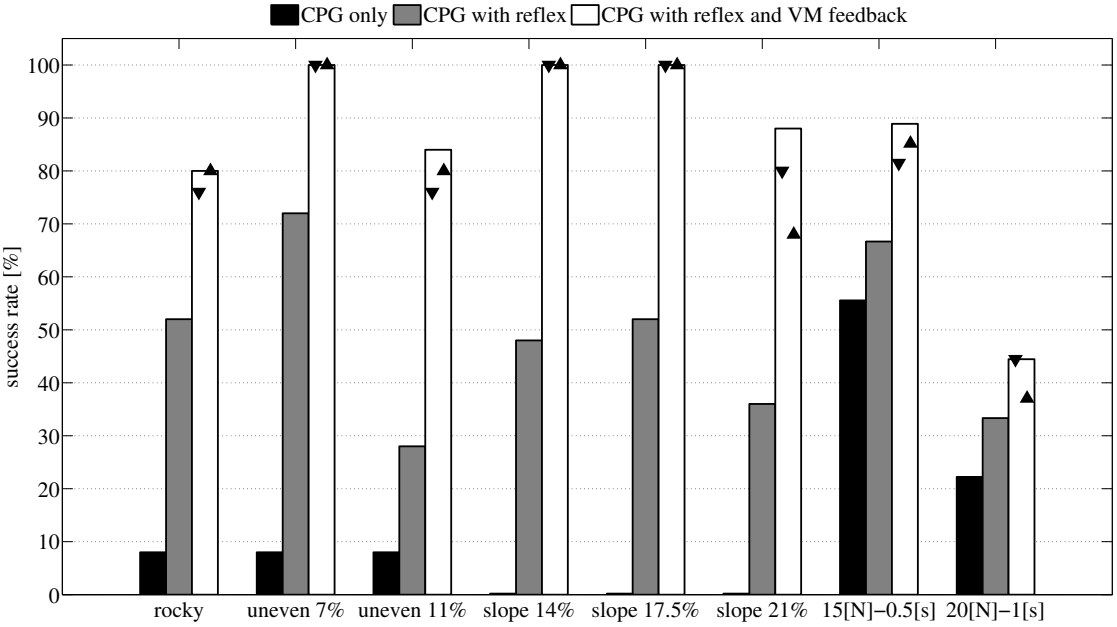


Figure 7.3: Results for the experiment scenarios. The open-loop gait is mostly unable to successfully pass the scenarios, other than 15[N] pushes where a 56% success rate is obtained. The results for activating the reflex mechanism are consistently better than the open-loop control, and similarly, the results obtained by additionally activating the posture controller (VM) is consistently better than the reflex-only case. The ▲ and ▼ markers on the white bars are for controllers with ±20% posture controller gains, as a measure of the sensitivity of the controller to the choice of its parameters. We additionally present results for a 20[N]-1[s] pushes scenario (column 8) where the pushes are just too big to be handled. Nevertheless, the application of the closed-loop controller improves the results even for this case.

rected (since there is no feedback). In case of the slopes environments, the robot stumbles in transition to the slope, or the trunk's pitch angle gradually increases and consequently leads to a fall. When facing external pushes, pitch and roll angles are not continuously corrected and robot falls for about 44% of the times.

In contrast, the closed-loop control is successful in most of the scenarios. In case of the rocky environment, the robot is successful for 20 out of 25 experiments. Normally the robot can go over the rocky terrain with minor difficulties. But since there is no anticipation, there are cases where the robot is greatly disturbed and needs to reactively correct the attitude. Uneven terrain is passed with a good success rate. For uneven terrain of about 7% of leg length variation, all the experiments where successful. The performance starts to degrade at higher terrain variations, as the momentary corrections are not sufficient to correct the attitude of the robot.

Closed-loop controller handles slope environments with a 100% success rate for $14-17.5\%$ slopes. Fall cases start to happen after 21% slopes. The attitude controller forces the robot to have a horizontal attitude w.r.t. world coordinates, and this has positive and negative effects

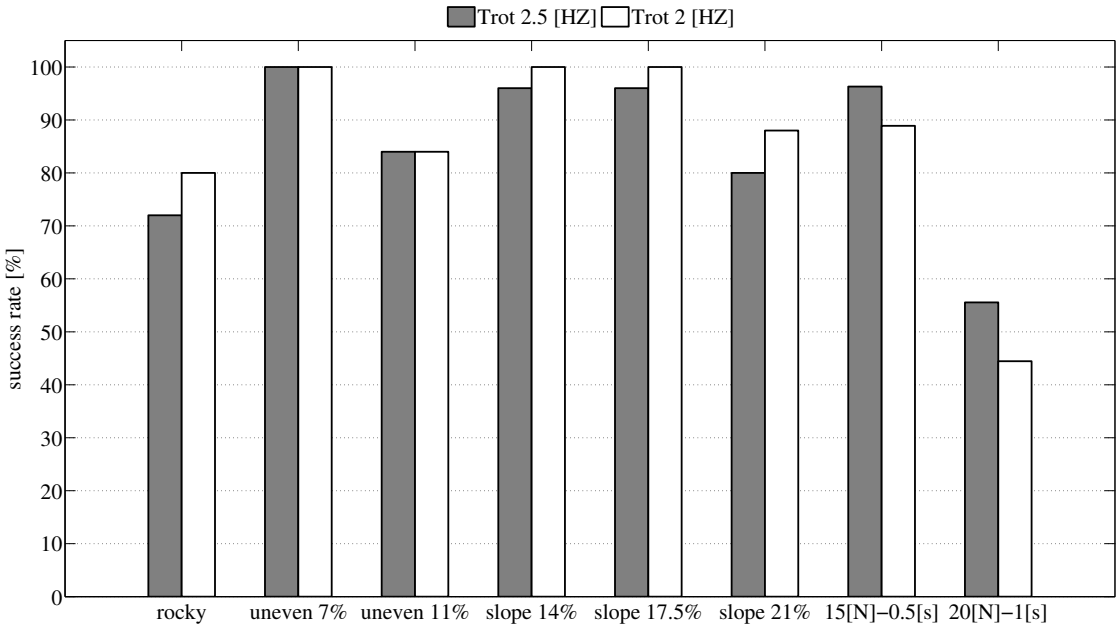


Figure 7.4: Results of the experiment scenarios for a faster trot gait at 2.5$[Hz]$ compared to the ones obtained for the gait used in section 7.1.3. There is not much variation in the results even though different gaits are used. The control gains are not tuned for the faster gait, and are kept as they were for the 2$[Hz]$ gait. The faster gait is more robust against external pushes.

when facing slopes. The positive is that the pitch angle is contained so it is less probable to fall backwards. The negative is that when the slopes become bigger, the generated torques for the attitude control also consistently become bigger and start to alter the CPG plan needed for forward progression. One would use the robot orientation along with the posture of the contact legs to estimate the slope, and use this to regulate the attitude controller. This is done by (SE) described in Chapter 6, and is applied for the experiments with the Oncilla robot, which are presented in the next section.

There were 3 fall cases out of 27 runs in 15[N]-0.5[s] pushes scenario. In two of the fall cases the external push was backwards and sideways, and only sideways in the third fall. We additionally tested our controller with external pushes of 20[N]-1$[s]$ which are very big for the robot's weight. Nevertheless, the performance of the closed-loop controller is twice better than the open-loop controller facing these external pushes (please see the last column in Figure 7.3).

We also tested the closed-loop controller with ±20% posture control gains as a partial measure of robustness to parameter changes. The results for these tests are presented with ▲ and ▼ markers in Figure 7.3. The results have not changed for less difficult setups like $uneven$ 7% or $slopes$ 14 – 17.5%. Performance starts to be sensitive to the choice of parameters at rougher terrains, since it makes sense to assume that the basin of attraction of the whole system (forward dynamics plus the closed-loop control) is naturally smaller in a more irregular environment.

7.1.4 Additional Test: A Faster Gait

To be able to show the generality of the introduced control methodology, we ran the experiment scenarios with another trot gait working at 2.5 [Hz] and a speed of about 0.9[m/s] which is 3[BL/s] (about 50% faster). We reused the gait parameters given in Table 7.1, changed the locomotion frequency to 2.5[Hz], and also added a virtual force of 5[N] pulling the robot in forward direction to obtain a faster locomotion speed. The control gains are kept the same as the ones in Table 7.1. The results for the experiments with this gait are illustrated in Figure 7.4. As expected, the obtained success rates are still acceptable, and comparable to the ones obtained in Figure 7.3. There is a minor drop in overall performance since the control gains are not tuned for this new faster gait.

7.1.5 Additional Test: Turning

Voluntary turning can be acquired by giving constant sideways virtual forces like the ones used for direction control. We could obtain a maximum of 90[deg/s] turning rate without greatly disturbing the robot. For this maximum turning rate we have $k_{trn} = 50$. This maximum turning is acquired at a locomotion speed of more than 2[BL/s] and a minimum turning radius of about 0.4[m]. Figure 7.5 depicts snapshots of a fast turning.

7.2 Oncilla: Position-controlled Quadruped

The second set of experiments in simulation are done using the Oncilla robot. The forward dynamics physics simulation is done using the Webots commercial software (with customized physics plugins to be as close as possible to the robot), and interfaced using the AMARSi software architecture [Soltoggio and Steil, 2012, Nordmann et al., 2013]. Both physics simulation and control loop are working at 500[Hz] (2[ms] timestep).

The (simulated) robot is a lightweight quadruped with passively compliant legs. The robot weighs 3.9Kg, the standing hip height is about 0.18[m], the distance between the shoulder/hip axes is 0.215[m] ipsilaterally , and 0.128[m] contralaterally. Each leg follows a three segmented pantograph mechanism, keeping the first and third segments parallel. All actuation is done proximally, so the legs are low-inertia. hip AA and hip PR joints are controlled on their motor axes, and the knee FE joint is controlled using a cable-clutch mechanism, actuated near the shoulder/hip point. Because of the parallel mechanism, the range of motion does not allow for singular configurations (e.g. a fully stretched leg). A more detailed description of the robot is given in Chapter 8.

All the results which are reported in the following are for the simulated robot locomoting with a trot gait with a forward speed of about 0.4[m/s], more than 1.7[BL/s].

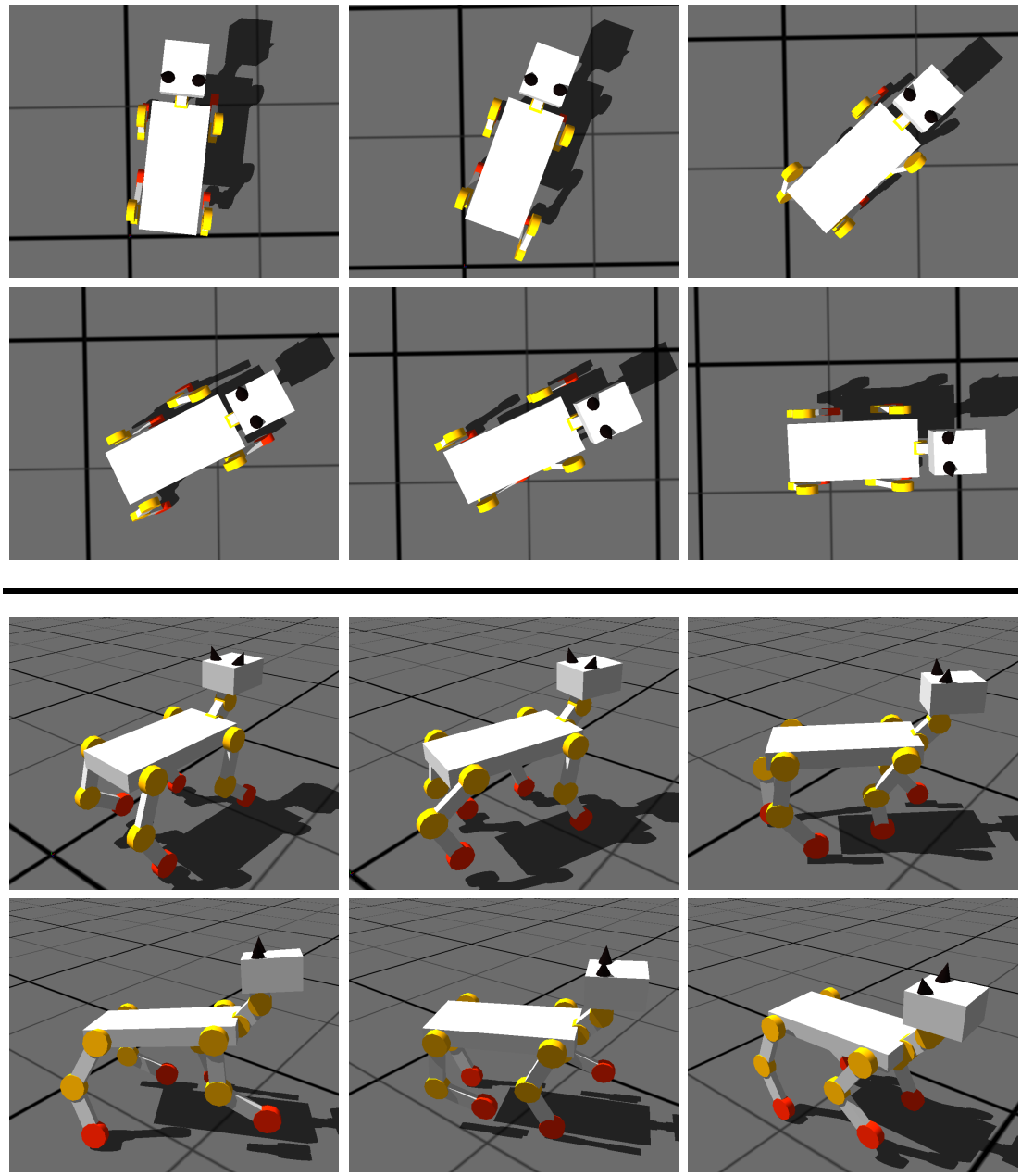
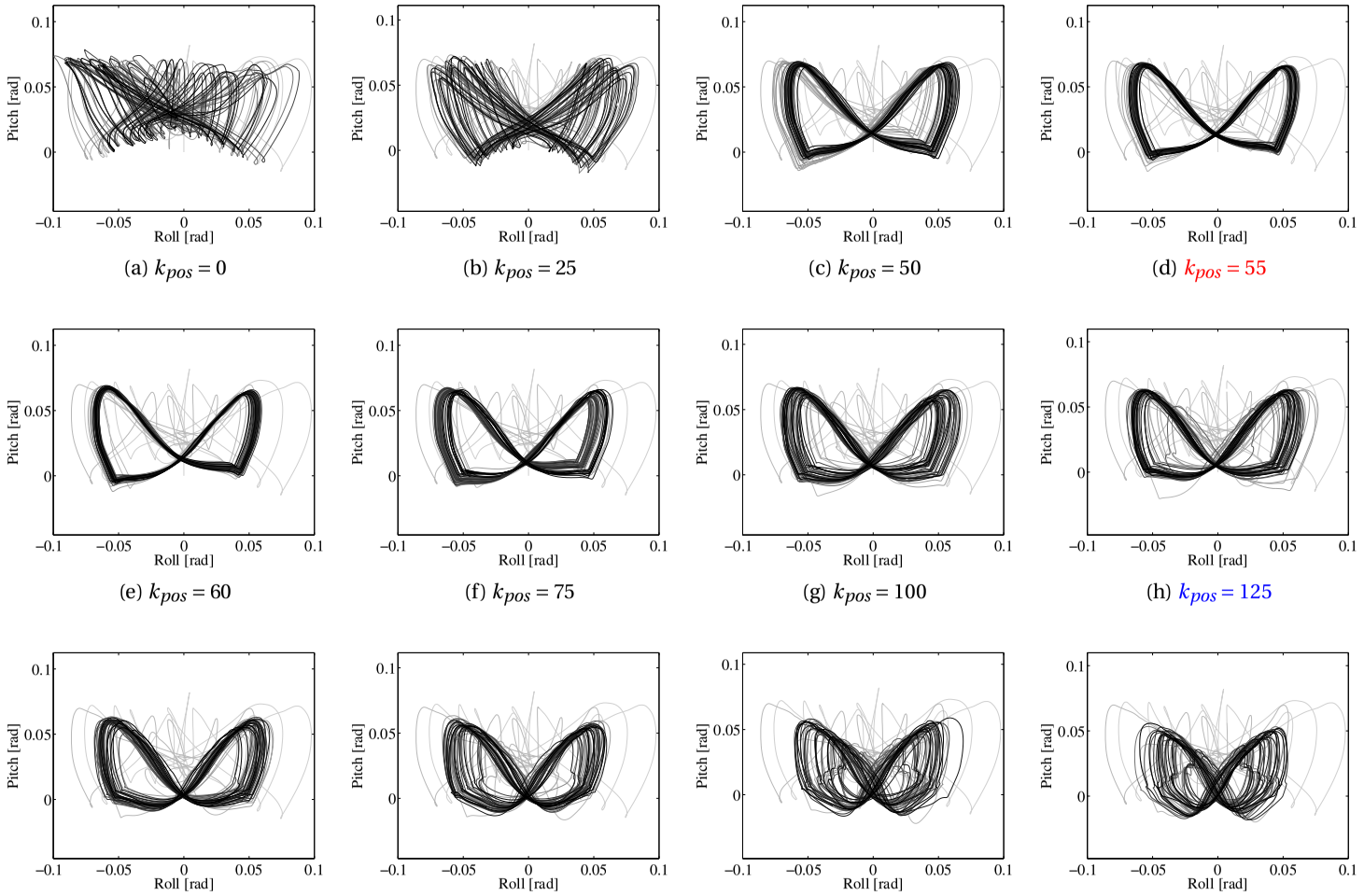


Figure 7.5: Snapshots of Ghostcat turning. The snapshots are taken every 0.2[s]. The robot turns with $k_{trn} = 50$ for 1[s]. A turning rate of about 90[deg/s] is obtained. The fast turning does not disturb locomotion, and the robot continues to normally locomote afterwards.

7.2.1 Control Parameters

Control parameters are chosen as described in Chapter 6. For the experiments presented, we have not used the foot trajectory regulator, and $f_i(.)$ limit cycles have been tuned manually, as same as for the Ghostcat (with the difference that a phase lag of $\frac{\pi}{3}$[rad] is introduced between hips and knees). For the angle-of-attack control, the hip offset strategy is used. To summarize,



(a) $k_{pos} = 0$ (b) $k_{pos} = 25$ (c) $k_{pos} = 50$ (d) $k_{pos} = 55$

(e) $k_{pos} = 60$ (f) $k_{pos} = 75$ (g) $k_{pos} = 100$ (h) $k_{pos} = 125$

(i) $k_{pos} = 150$ (j) $k_{pos} = 175$ (k) $k_{pos} = 200$ (l) $k_{pos} = 225$

Figure 7.6: Roll and pitch variations (RPV) for different posture control gains. A good value of for k_{pos} stabilizes the RPV and make it more periodic, however excessive increase of this gain can have a counter effect. The optimal value seems to be about $k_{pos} = 55$ (red), but we use a larger gain of $k_{pos} = 125$ (blue) for the experiments to have a stronger resistance to perturbations.

the control parameters are set to $k_{pos} = 125$, $k_{rfx} = 50$, and $k_s = 1.25$.

We would like to address again the effect of the posture control gain on the RPV when walking on flat terrain. Figure 7.6 illustrates the effect of increasing k_{pos} by small steps. For $k_{pos} = 0$ the RPV is actually not very periodic and drifts and changes size. By closing the loop, even with a small k_{pos}, the drift is gone. We find a nearly optimal value at $k_{pos} = 55$. This value is for flat terrain, so we choose a larger value of $k_{pos} = 125$ for the rough terrain experiments. Please note that excess increase of k_{pos} starts to be destructive.

7.2.2 Rough Terrain Locomotion

Three different scenarios were used to evaluate the performance of applying our proposed control architecture on Oncilla (Figure 7.7):

- Randomized uneven terrain, 12% of the leg-length height variations (max local slope = ±20%);
- Downward step, 20% of the leg-length height;
- 20% downward slopes.

Each of the above scenarios is repeated 25 times from different initial conditions (robot is placed in different initial positions w.r.t. the rough terrain). Each experiment is ran for 20 seconds from which the first $5-8$ seconds is used for initialization (unperturbed). The same gains as described before are used for all the scenarios, and we do not change or re-tune the gains for different scenarios. The controller does not have any kind of prior information about the environment and the perturbation scenario.

The overall results of the rough terrain locomotion scenarios are shown in Figure 7.8. A CPG-only control was partially successful on the **randomized uneven terrain**. As from our study on the stiff-by-nature quadruped Ghostcat, we were expecting the open-loop controller to perform poorly, however, a 56% success rate was obtained. This partial success is due to the passive compliance in the joints, which passively flexes/extends the legs in reaction to the environment, and moderately self-stabilizes the roll and pitch oscillations. This is similar to what is reported in [Spröwitz et al., 2013]. Nevertheless, the posture control and reflex mechanisms are needed for a better performance. As Figure 7.8 shows, a 96% success rate is obtained by applying the whole closed-loop control.

The CPG-only control was mostly unsuccessful in the **downwards step** scenario and only 20% of the trials were successfully passed. In contrast, the CPG control with reflex and posture control feedbacks successfully passed the trials. The leg extension reflex is very important for this scenario, as it compensates for the missing contact at the step down. The posture control mechanism comes into play after the step where the body oscillations, induced by the perturbation caused by the step, should be stabilized.

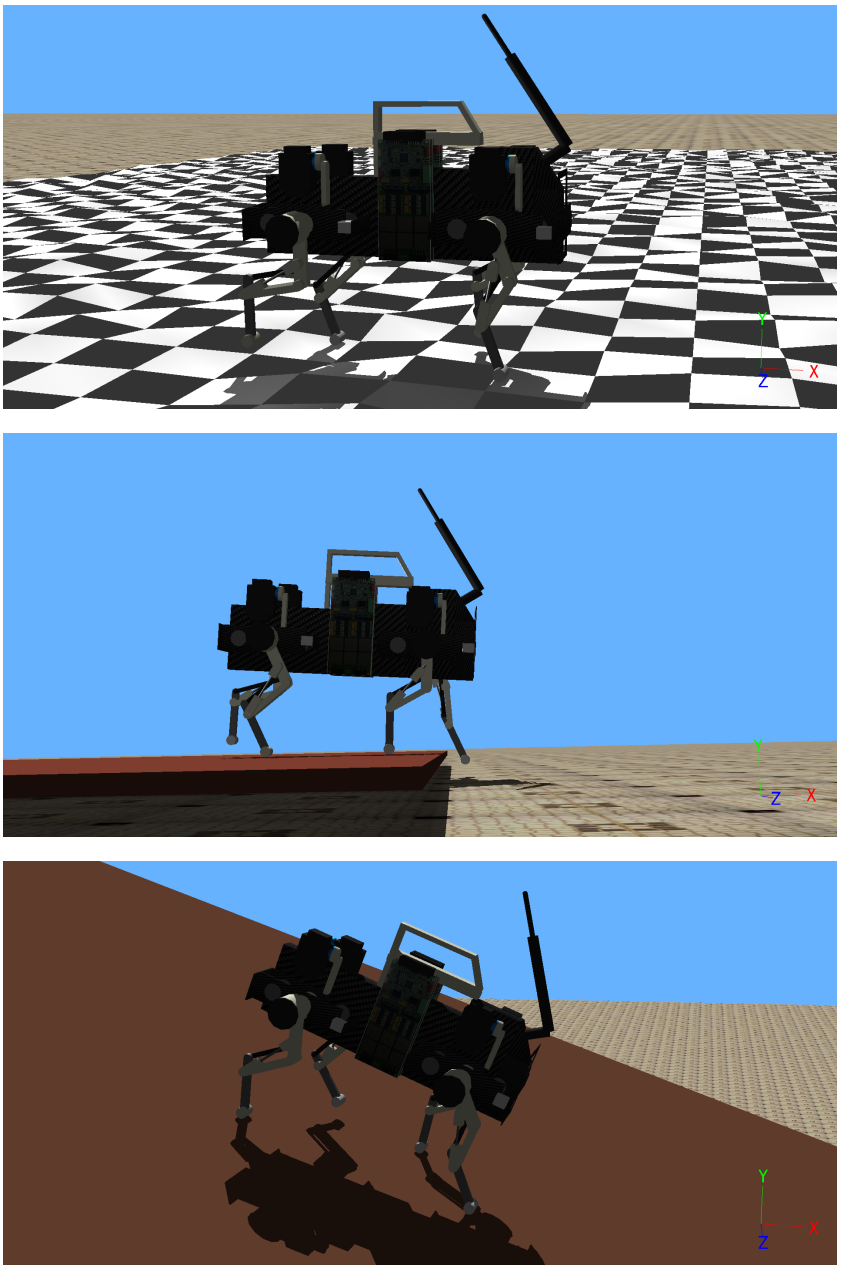


Figure 7.7: The simulated Oncilla, and blind rough terrain locomotion scenarios. *Left,* randomized uneven terrain with 12% of leg length variations. *Middle,* downwards step, 20% of leg length. *Right,* 36.5% downwards slope.

None of the **slope** experiments were successfully passed using a CPG-only control. Again, both reflex and posture control mechanisms are crucial for success in this scenario as they prevent stumbling, compensate for missing contacts, and keep the body roll and pitch oscillations contained. We additionally tested our control method against 36.5% (20 degrees) downward slopes, which are quite difficult for a blind controller. We realized that a fine tuning of the reflex gains is needed for this case ($k_{rfx} = 120$ for the leg extension reflex and $k_{rfx} = 50$ for the

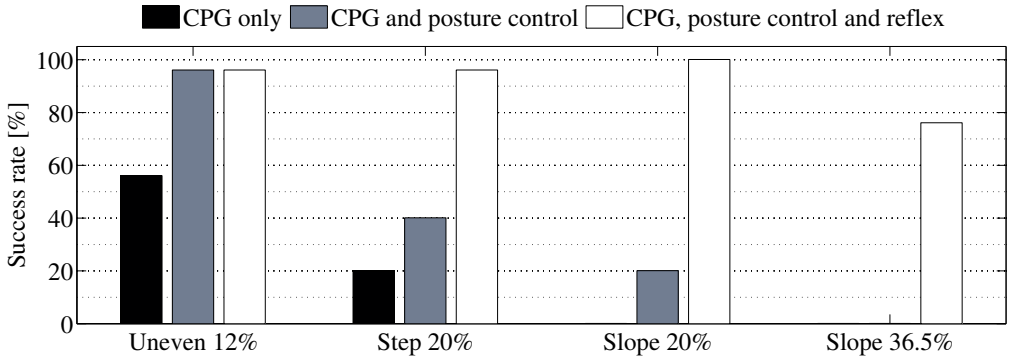


Figure 7.8: Performance of the *CPG only*, *CPG + posture control* and *CPG + reflex + posture control* controllers on unperceived rough terrain. The *CPG only* control is partially successful on the uneven terrain because of the compliance, and the compliance fulfills the role of a weak reflex mechanism. A much better performance can be obtained by adding the posture control module. Only the complete control (*CPG + posture control + reflex*) is successful in all of the scenarios. We additionally test with an extra scenario, downward 36.5% slopes, and the robot was successful in 19 out of 25 trials. In all the scenarios, a consistent increase of the performance is observed by adding the posture control and reflex modules.

stumbling correction reflex), and then the robot can pass this scenario successfully (Figure 7.9, black columns). This means that, having the prior knowledge about the environment, the reflex gain can be coupled to the slope inclination.

To show the importance of the reflex and posture controller modules, we ran the control with different reflex and posture control gains. Figure 7.9 shows the performance of the control with reduced posture control and reflex gains. A lower reflex gain ($k_{rfx} = 25$) lowers the performance in case of the steps and the steeper slopes, and a reduced posture control gain ($k_{pos} = 60$) affects the overall performance.

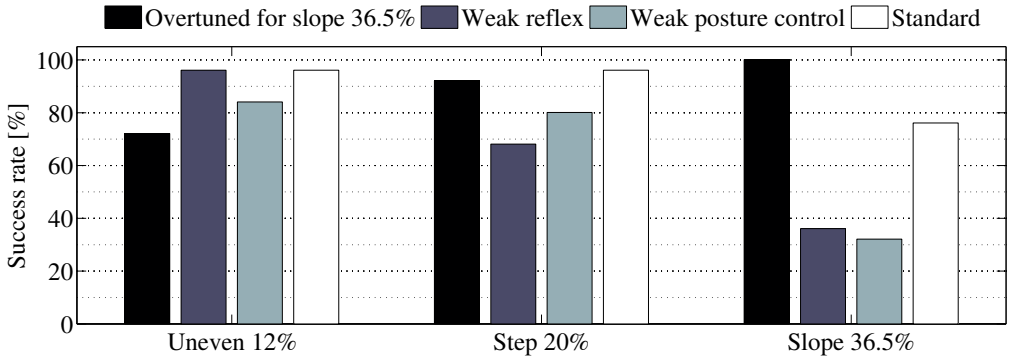


Figure 7.9: Variations of the control parameters. Black) The control can be overtuned to perform well on 36.5% slopes, but overtuning will affect the performance in the other scenarios. Purple) A weaker reflex ($k_r = 25$) leads to lower performance in step and slope environments. Azure) Weaker posture control gain ($k_{att} = 60$) affects the whole performance.

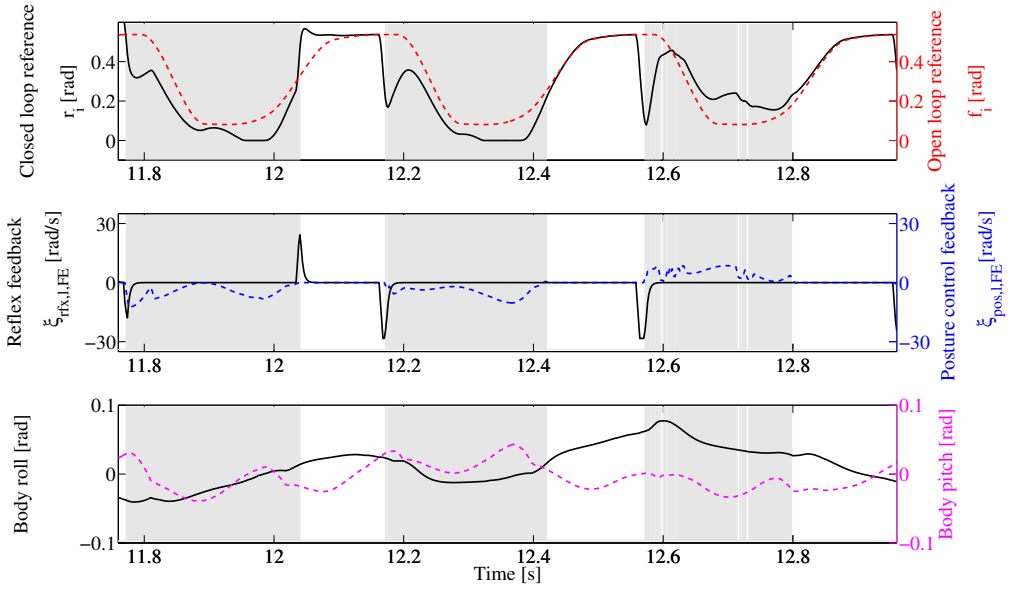


Figure 7.10: Control signals for an example run on uneven terrain. The signals are for the control of a hind knee (FE) joint. Posture control feedback continuously adjusts the control reference. Stumbling correction reflex is activated just after $t = 12$s and the leg extension reflex is activated two times before $t = 12.2$s and $t = 12.6$s. Please note that positive values for the FE joint relate to flexions (shortening of the leg length).

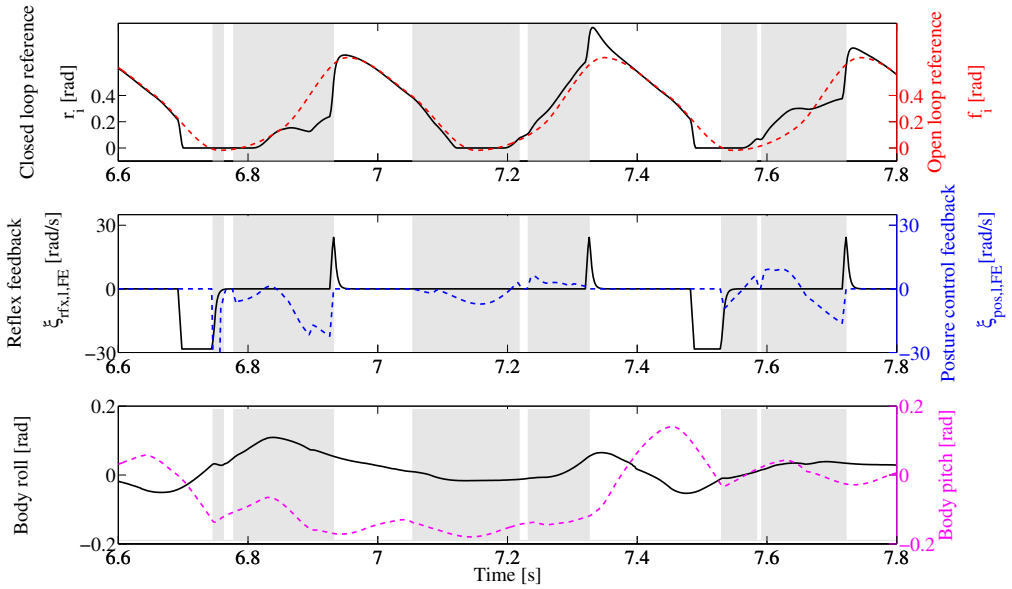


Figure 7.11: Control signals for the moment that the robot goes over a downwards step (fore knee). The step occurs around $t = 6.7$s, and causes a missing contact state, and a leg extension reflex is activated until a contact is sensed. At that moment, since the robot is pitched, the posture controller is strongly activated to correct the body posture. Since the robot is pitched the fore leg drags on the ground in the beginning of the two next swing phases, and stumbling correction reflexes are activated. The hind leg comes down the step around $t = 7.4$ (see the correction in the pitch angle), and causes a small impact which slightly lifts the front of the robot, and another leg extension reflex is activated in the fore knee to acquire ground contact. The reflex and posture control feedbacks are damped in the beginning of each swing phase (white background), and the system resynchronizes.

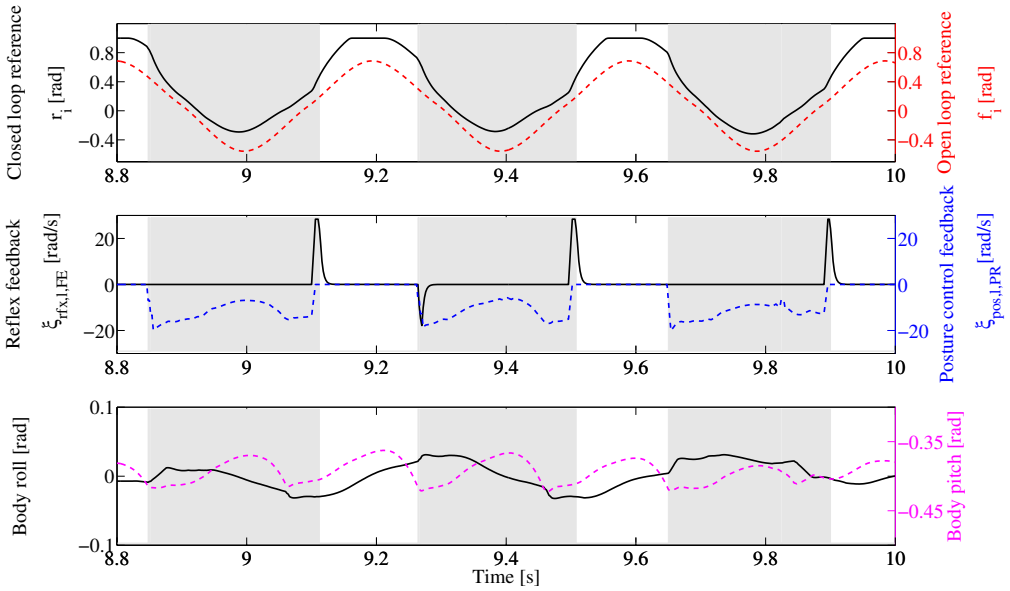


Figure 7.12: Control signals for a saggital fore hip (PR) in a 36.5% slope scenario. the angle of attack control adds an extra hip offset to the r_i joint angle reference (there is a soft joint limit at $r_i = 1$). The activation of reflex and posture control feedbacks is quite repetitive, which means that the robot is in a new limit cycle behavior adjusted for the slope. The same gains as for all other scenarios are used.

7.2.3 Control Signals

Figure 7.10 illustrates the evolution of the control signals over time for locomotion on the randomized uneven terrain[1]. The illustrations are for three stride cycles of a hind knee (FE) joint. Posture control feedbacks continuously adjust the joint angle reference, while reflexes are short term and for fast corrections. The CPG state r_i converges back to the encoded limit cycle f_i in each swing phase (white background), and the effect of the feedbacks are damped since there is no ground contact, hence the control system resynchronizes.

Figure 7.11 illustrates example control signals at the moment of a step down, for a fore knee (FE) joint. Again, the posture control feedbacks are continuously adjusting the joint angles reference, while the reflexes are quick and short term. Please refer to the caption of Figure 7.11 for details.

Figure 7.12 corresponds to locomotion on a downwards 36.5% slope. The signals are for the sagittal hip (PR) joint of a fore leg (since there are no reflexes implemented for the PR joint, ξ_{rfx} is given for the FE joint of the same leg). As the figure shows, the body rotations are stabilized, and the activation of the feedbacks are repetitive over the cycles. The effect of the angle-of-attack feedback is also visible in the offset added to the r_i reference.

[1] For Figures 7.10-7.12, left and right y-axes correspond to the solid and dashed lines respectively. For example, the top subplot in Figure 7.10 contains two trajectories, closed-loop reference r_i with black solid lines, and open-loop reference f_i with red dashed lines. The y-axis quantities are different for each subplot.

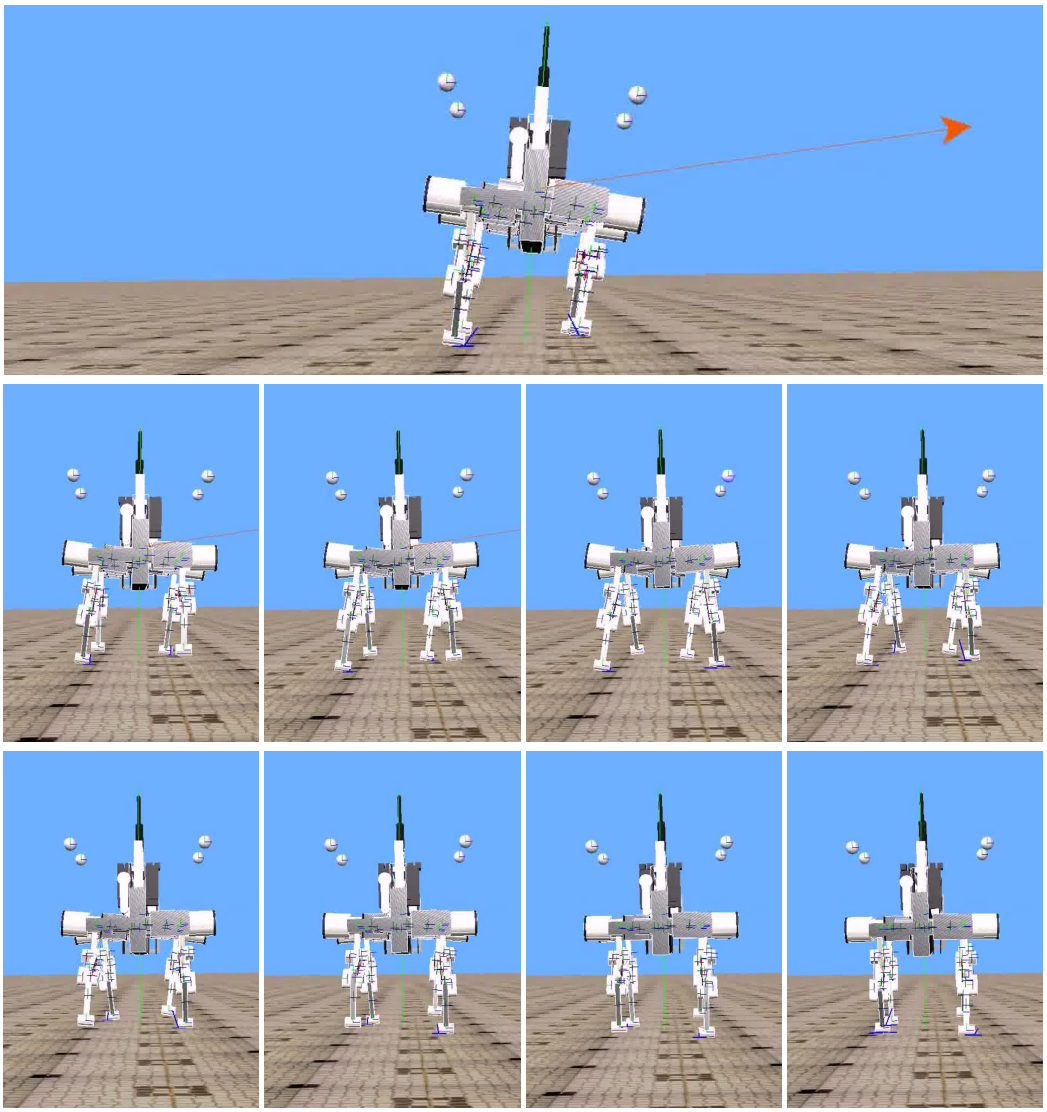


Figure 7.13: Sidestepping after a lateral perturbation. *Top,* the robot is pulled by an external force, and it starts to tip over. *Snapshots* of the lateral stepping taken every $\frac{1}{30}$[s].

7.2.4 Additional Test: Lateral Push Recovery

Figure 7.13 shows the activation of the lateral stepping reflex (LSR) after the robot is pulled by an external force. When the force is applied, the robot starts to tip over. As the amount of the lateral acceleration is excessive, LSR is activated to take sidesteps and capture the body. For this figure, the robot can successfully dampen the perturbation in less than one stride cycle.

7.2.5 Additional Test: Asymmetric Load Carriage

We tested our controller with the task to carry asymmetric loads. We place a load 0.08[m] left side of the trunk while there is no load on the right side. If the posture controller is tuned

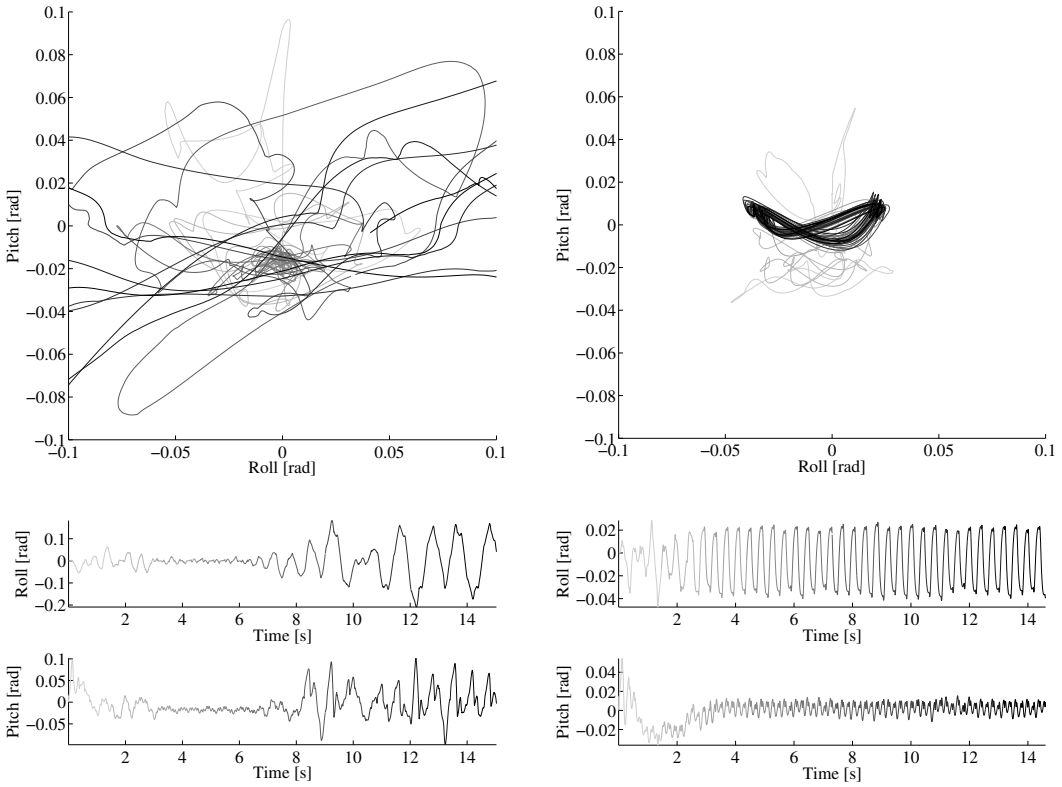


Figure 7.14: RPV comparison of the open-loop and the closed-loop controllers, for the case of asymmetric 0.1[Kg] load. Open-loop control does not lead to a fall, but the robot is constantly perturbed, and the roll amplitude is magnified in some parts. The closed-loop controller keeps RPV in check and the robot smoothly locomotes. Note the the presence of the asymmetric load can be seen by the asymmetry in the roll (the infinity-like figure is inflated on the right).

off, the robot can carry a maximum of 0.1[Kg] of asymmetric load without falling. However, if the posture controller is turned on, the maximum asymmetric load is 0.4[Kg]. Figure 7.14 compares the RPV of the open-loop and the closed-loop controllers, for the case of asymmetric 0.1[Kg] load.

7.2.6 Additional Test: Speed Control

Figure 7.15 illustrates the quality of the speed control strategy explained in Chapter 6, by the change of the step length. For the illustrated examples, the gait is fixed to be a trot gait (even for slow speeds), with a fixed duty factor of $d = 0.5$. The actual speed of the robot is calculated taking into account the displacement of the trunk from one cycle to the next, and the cycle duration. At the zero speed, the robot is performing an in-place trotting. We see that as long as the request in speed change is not abrupt, the robot can nicely follow the desired speed profile. For abrupt changes, like in the fourth example in Figure 7.15, it takes about 1[s], equal to 2.5 stride cycles, to reach the desired speed.

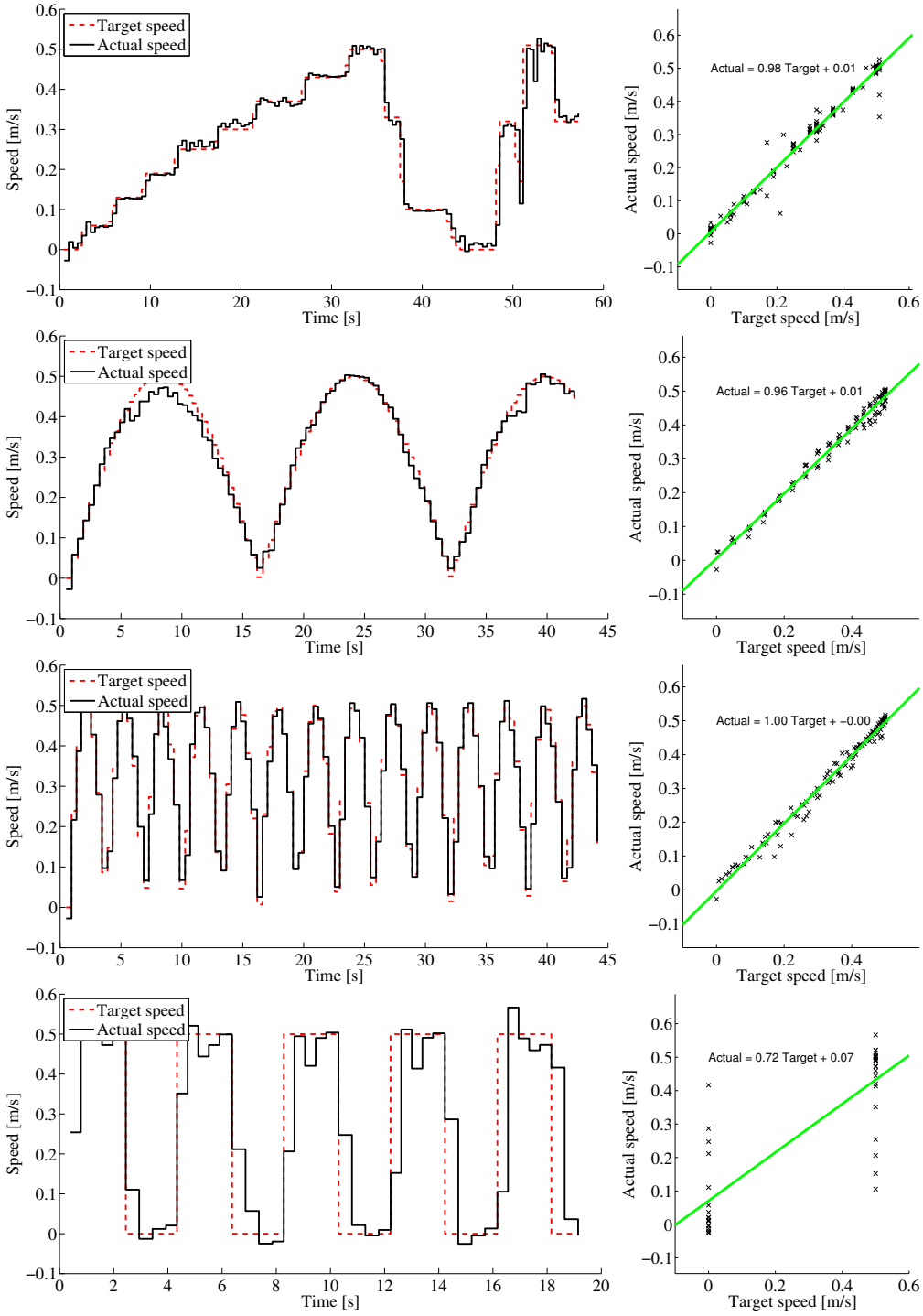


Figure 7.15: Speed control examples. *First from top,* a typical case where the robot gradually speeds up, quickly brakes, and quickly speeds up again, with an abrupt brake in between. *Second,* slow speeding up and down. *Third,* quick speeding up and down. *Fourth,* requests for abrupt speed changes. We see that other than the *fourth* example, the speed control is precise. For the *fourth* example, it takes about 1[s] in average to reach the desired speed.

7.3 Summary

We presented the results of applying the proposed control architecture to locomotion of simulated quadrupedal robots. We used Ghostcat, a stiff-by-nature quadruped, and Oncilla, a passively-compliant one for the experimentation.

We showed how closing the loop and adding the feedback mechanisms cleans up the role-pitch variations (RPV), removes the drift, and make RPV more periodic. Closing the loop can even stabilize gaits which are open-loop-unstable. The interesting point is that the relationship between the posture control gain and the degree of periodicity of the gait is quadratic: increase of the posture control gain first improves the performance, reaches the optimal point, and further increase has a negative effect, as shown in Figure 7.6

Utilizing the proposed control architecture, both quadrupeds could blindly locomote on moderately difficult rough terrain. Though the robots did not have any information about the geometry of the environment (or even its type), and same control parameters were used through all the experiments with one robot, and led to high success rates. We showed that a CPG-only control is barely enough for blind rough terrain locomotion, and posture control and reflex mechanisms are necessary.

As a measure of control robustness, we showed that the proposed control framework is not overly sensitive to the choice of control parameters, and having different gait parameters or lower/higher feedback gains still leads to acceptable results. We also shows that if the robot knows about the environment in-advance, like in 36.5% slope experiment with Oncilla, the control parameters can be adjusted for an even better performance.

We showed that the proposed control architecture can also be used for other skills like fast turning, carrying asymmetric load, and speed control. We demonstrated turning as fast as 90[deg/s], carrying asymmetric loads of more than 10% of the body weight, and precise speed control. These are all done without any additional tuning of the parameters, and come straight out of the capabilities of the introduced control framework.

By comparing the stiff-by-nature and compliant quadrupeds, we could see that compliance helps the open-loop control to be more successful (compare Figure 7.3 and 7.8 for the results of locomotion on uneven terrain with a CPG-only controller). Compliance can take care of small perturbations by passive flexion/extension of the leg. Nevertheless, the closed-loop controller is needed for successful blind rough terrain locomotion.

Finally, we demonstrated that the same control architecture can be used for two quite different platforms. The main point of deviation between the control on the two platforms is the way posture control is implemented: Ghostcat utilizes a modular posture controller through the application of virtual forces, and Oncilla uses a whole-body posture controller based on virtual velocity feedbacks. Nevertheless, results presented in this chapter show that the integration of CPGs and posture control can be done both at the CPG level (Oncilla), or at the motor

control level (Ghostcat). It is an interesting question to explore whether the posture control feedback in animals passes through CPG circuitry, directly modulates the motor neuron and muscle activities, or a combination of these. This question can also be explored at a high-level of abstraction utilizing the proposed control architecture, by simultaneously using posture control feedback at both CPG and motor control levels, and this is left for future research.

8 Hardware Experiments

千里之行，始於足下 *(A journey of a thousand miles begins with a single step).*

Lao-Tzu

Publication

Parts of the material used for the hardware description (Section 8.1) presented in this chapter are taken, with permission, from:

– Alexander Spröwitz, Alexandre Tuleu, Mostafa Ajaoolleian, Massimo Vespignani, Rico Moeckel, Peter Eckert, Michiel D'Haene, Jonas Degrave, Arne Nordmann, Benjamin Schrauwen, Jochen Steil, and Auke J. Ijspeert. Oncilla Robot: A Compliant, Versatile, Open-source Quadruped Robot with Pantograph Legs. *Prepared for submission.*

My contributions to this publication are the implementation of the high-level controller, tuning of the gaits, and performing hardware experiments.

The previous chapter presented the results of applying our proposed control architecture on simulated platforms. We use the Oncilla robot to validate the simulation results on a hardware platform. As the Oncilla robot is position controlled, we only use the position-control approach to the posture control.

We first describe the hardware properties of the Oncilla robot. Then, we explain the points of deviation from simulation. Lastly, we will demonstrate the results obtained from the hardware experiments.

8.1 The Oncilla Robot

The Oncilla robot (Figure 8.1) is a quadrupedal platform made within the framework of the AMARSi EU project (Chapter 2). Oncilla weights 5[Kg] including a three-cell Corally 4500[mAh] 45C LiPo battery (1.1[Kg] heavier than the simulated robot). The dimensions of Oncilla are about $0.18 \times 0.22 \times 0.14$[m] (height/length/width).

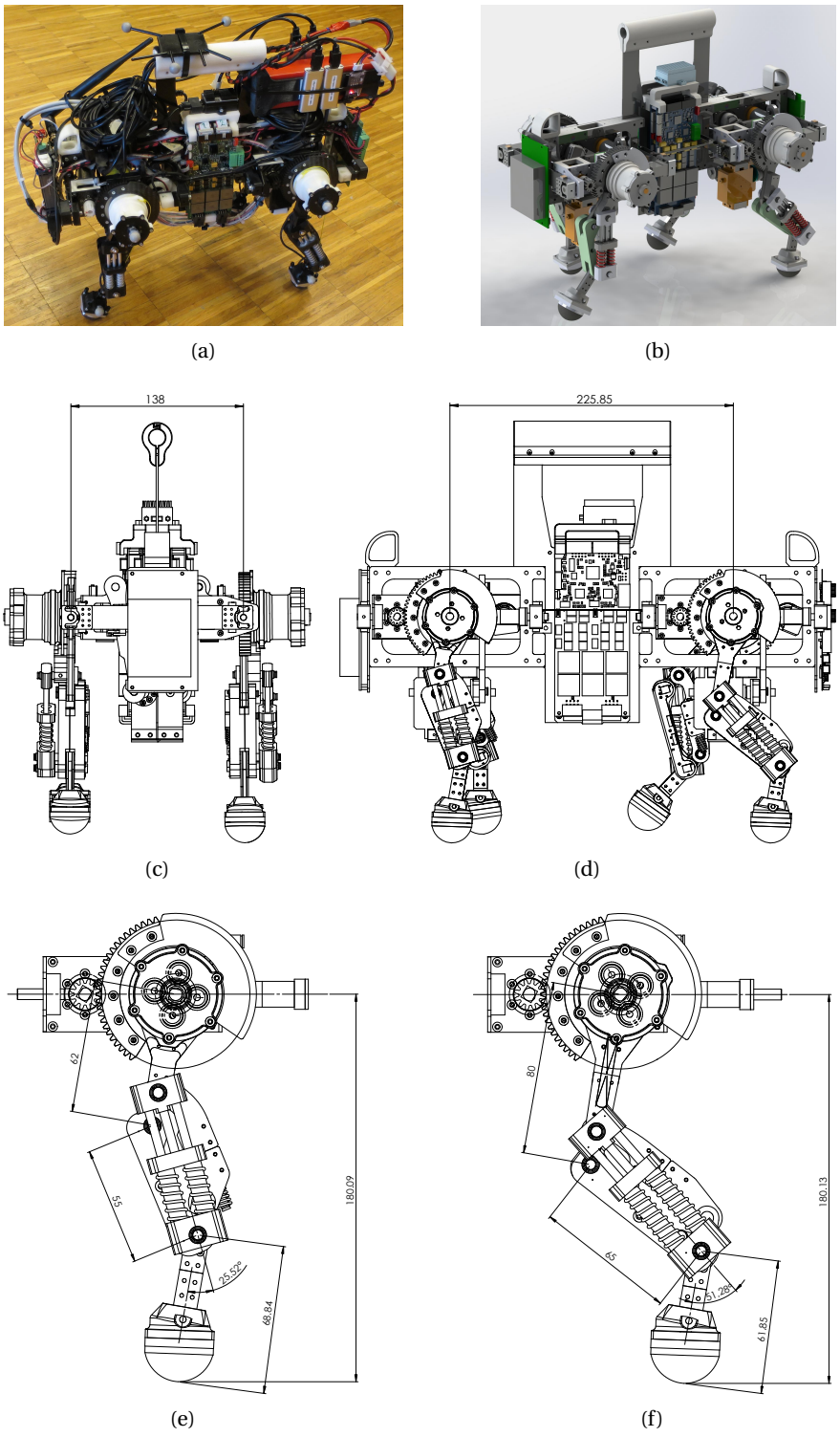


Figure 8.1: Oncilla robot (a) the hardware robot, (b) CAD model, (c) front view, (d) side view, (e) fore leg, (f) hind leg. Oncilla-robot is a tetherless, quadruped robot. It features three active degrees of freedom per leg. The three segments per leg are mounted into a pantograph configuration, with a ”relief spring“ replacing one parallel link of the pantograph.

Each leg includes five degrees of freedom, namely hip AA, hip PR, knee FE, ankle FE, and heel FE. The movement of knee and ankle joints are coupled through a four-bar pantograph mechanism. There is a linear spring of about 3500[N/m] placed diagonally inside the four-bar mechanism. The fourth bar is also replaced with another linear spring, making an "open" pantograph. The heel joint is equipped with a torsion spring and is passively compliant.

The actuation for the hip AA is done using Kondo KRS-2352 servo motors, and is transmitted to the axis of rotation using a lever. The hip PR is actuated by a Maxon M110524 brushless motor on the axis. Another Maxon M110524 brushless motor is placed proximally near the hip axis and actuates the knee FE (and ankle FE) through a cable-clutch mechanism. Thus the flexion of the leg is active while extension is done passively by the diagonal spring. If there is no ground contact, knee FE and ankle FE joints will have the same angles.

The robot is equipped with magnetic joint encoders of hip PR, knee FE and ankle FE joint, however there is no joint angle sensing on hip AA or heel FE. We use a Microstrain 3DM-GX3-35 IMU for the sensing of trunk rotations and accelerations. The IMU is capable of providing precise measurements of roll and pitch, but the yaw measurement can drift.

For contact sensing, we have tried four different approaches:

1. Using strain-gauge load-cells located at the hip axis. A reflection of the ground reaction forces can be seen on these load-cells, however the leg dynamics are also reflected on the sensor values. We could not easily detect a ground contact using these sensors;
2. Monitoring the difference between the knee and ankle encoders. As explained before, the knee and ankle joints are coupled through the open pantograph mechanism. This means that load can compress the leg and give similar angles to the knee FE and ankle FE. However, since the pantograph is opened by the replacement of a linkage with a spring, the flexion of knee FE and ankle FE are going to be slightly different. This can be used to detect contact. We decided not to used this method as it only senses the contact if the contact force is large enough, and also the relation between the extra ankle flexion and the force is dependent on the leg-ground angle;
3. Using pressure sensors embedded into 3D printed feet[1]. This approach was quite successful and the binary contact sensing was possible for most of the cases. However, depending on the placement of the pressure sensor inside the 3D printed piece, touching some parts of the feet was not sensed. Thus, the stance phase was not 100% captured, and also the feet were not sensitive in front to detect stumbling;
4. Using 3D force sensors. We used Optoforce 3D force sensors which use infrared light to detect the smallest deformation in the shape of the outer deformable surface of the sensor. These sensors are used for contact sensing in this thesis. Forces in the front direction are used for the stumbling detection, and the norm of horizontal and vertical forces are used to detect ground contact. As Figure 8.2 depicts, we have mounted

[1] The credits for these sensors go to Massimo Vespignani.

Figure 8.2: Optoforce 3D sensors. *Left,* the sensor mounted as a foot in replacement for old foot and heel joint. *Right,* cotton socks are used to protect the sensor against friction damage.

one sensor as the foot for each leg which replaces both the old foot and the heel joint. Moreover, since the surface of the sensors have a very high friction coefficient, we have covered the feet with cotton socks used for newborn babies.

All the control on the robot is done on-board. The robot is equipped with a RoBoard RB-110 which has a single 1[GHz] CPU and 256[MB] DRAM. The hip AA servo motors are directly connected to the RB-110, and custom made motor-driver boards are used for the control of the brushless motors. Custom-compiled Linux with real-time Xenomai kernel is used as the operating system. The control loop (including the high-level control) can be closed at best at 250[Hz]. We do most of the experiments at 200[Hz].

8.2 Deviation from Simulation

Most of the control done on the Oncilla hardware robot is the same as in simulation. However there are a number of issues which has forced us to have some modifications in the control. The list of issues include:

- There are no joint encoders on the hip AA joints, thus the calibration of these joints is difficult, and a perfect calibration is not always acquired. Due to this, the robot can move a little bit sideways while locomoting forward. Nevertheless, the effect is minimal (about 0.1[m] of lateral displacement for 5[m] of forward displacement);
- The range of motion for the hip AA joint is much smaller than in simulation, and is about ±5[deg]. This greatly limits the capabilities of the robot in handling lateral perturbations caused by an explicit external force or the ground geometry;
- The mechanical design of the legs have the springs on the outer side of the leg. This means that the leg, under the loads of the springs, slightly bends outwards, and the bending is bigger for stronger springs. Thus there has been a limitation of the maximum stiffness that could be introduced into the leg, and the hardware robot has a more compliant leg compared to the simulated one. This causes the passive extension of the knee FE joints to be slower;

- The hardware robot is about 25% heavier than in simulation. This, along the softer springs, have made the robot to be more crouched while locomoting;
- The yaw value of the IMU is not accurate.

Respective to the aforementioned list, we had three important modifications in the control of the hardware robot:

1. Since the range of motion for hip AA is very small, we leave out the hip AA joint from the posture control, and only use it for turning and sidestepping. If the hip AA joint is used for posture control, it quickly reaches the joint limits even on small lateral slopes;
2. Because of the excessive weight, we had to use a 50% smaller foot clearance to have a working gait. However, this makes the robot more sensitive to obstacles that can make the robot stumble;
3. Because of the inaccuracies in yaw sensing, we always use the straight rotation matrix (which excludes yaw) instead of the full rotation matrix sensed by the IMU. The heading of the robot is controlled by the operator.

8.3 Experiments

We now present the results obtained from the hardware experiments. This includes locomotion on the flat terrain with or without asymmetric load, lateral perturbation tests, and locomotion on uneven terrain.

8.3.1 Flat Terrain

For the flat terrain locomotion, we directly ported the controller from simulation to hardware, and it worked out-of-the-box (the movie is accompanied online). With further tuning of the gait parameters, we can go up to maximum forward locomotion speed of 0.7[m/s], more than 3[BL/s]. The same speed can be acquired for backward locomotion. Flat terrain locomotion is achieved by a CPG-only control, and feedback is not needed for this case. For the experiments in this thesis, we mostly use a speed of 0.4 to 0.5[m/s]. Snapshots of the forward locomotion are provided in Figure 8.3. As shown in the figure, the contact sensors can be omitted for flat terrain locomotion.

We showed in the previous chapter that the trunk Role-Pitch-Variations (RPV) can be improved by closing the loop and activating the posture controller. Figure 8.4 illustrates the RPV for open- and closed-loop controllers. The figure does not show a clear improvement of RPV. To better assess the periodicity of the gait, we collected several cycles of locomotion, stacked them together, and calculated the mean and standard deviation of trunk roll and pitch for several phase value (every $\frac{\pi}{10}$). Figures 8.5 and 8.6 show the mean and standard deviation of the roll and pitch values. We can now see that the variations over different phases of locomotion are smaller when the loop is closed. Moreover, if we calculate the average of all standard deviation

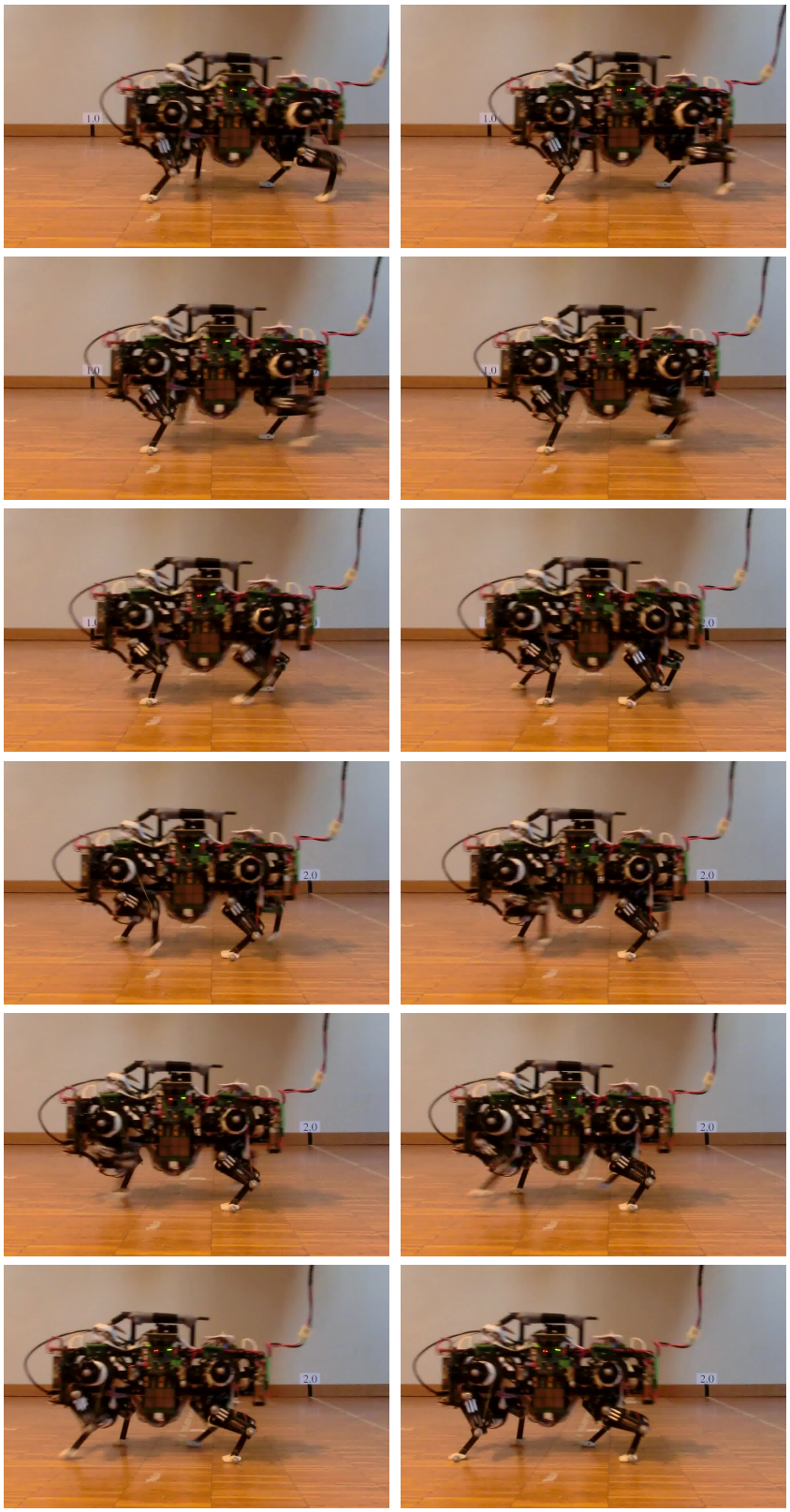


Figure 8.3: Snapshots of Oncilla trotting forward at about 0.5[m/s], taken every $\frac{1}{15}$[s]. The sequence is row-major from *top-left* to *right-bottom*.

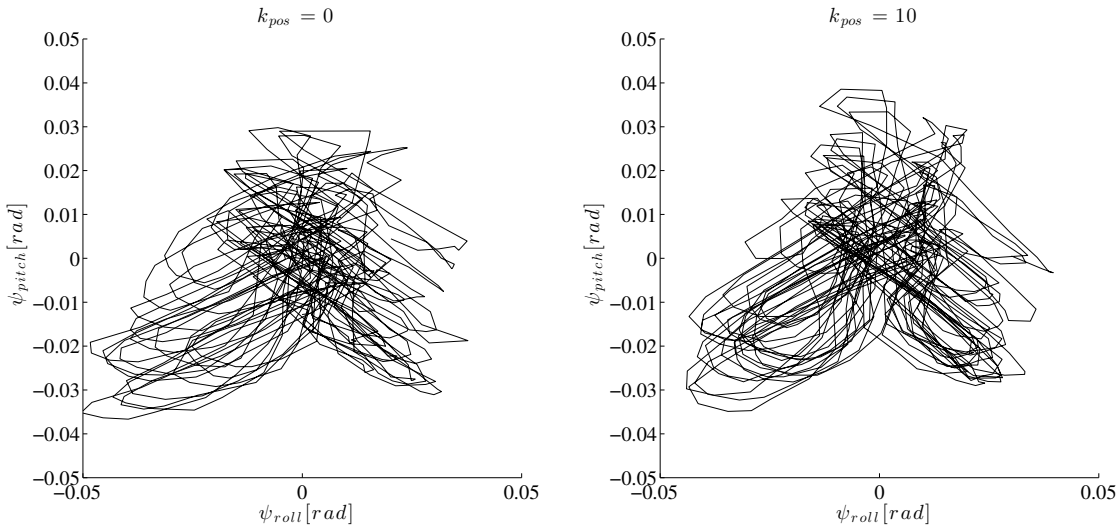


Figure 8.4: RPV for open- and closed-loop controllers for flat terrain locomotion. An improvement in periodicity is not obvious. Figures 8.5 and 8.6 analyze the details of this figure.

values, it shows that the roll variation is about 32% less, and the improvement is 13% for the pitch variations. Nevertheless, the improvement is not as visible as in the simulations. This is partially due to that fact the in the hardware legs cannot be perfectly calibrated and the weight distribution is not completely symmetric.

For the turning capability, we have implemented both the approaches to use the hip AA joint, or to scale the foot trajectory. The robot can easily turn in-place as fast as 45[deg/s], and a maximum turning rate of 90[deg/s] could be acquired, however with an extra care.

8.3.2 Asymmetric Load Carriage

To better show the capability of the closed-loop controller, we test the robot with carrying asymmetric load with both open- and closed-loop controllers. We use a load of 0.5[Kg] placed 0.3[m] right side of the trunk. This induces a continuous torque of 0.15[Nm] in the transverse plane. Figure 8.7 illustrates the roll and pitch of the trunk while carrying the asymmetric load, for both the open- and closed-loop controllers. When $k_{pos} = 30$, the average roll is about two times smaller than when $k_{pos} = 0$. This shows that the posture controller is constantly adjusting the roll toward the zero value. Moreover, the other important value in this experiment is the maximum roll value, which is about 0.18[rad] for the open-loop controller, and about 0.11[rad] for the closed-loop controller. This means that the open-loop controller is more in danger of falling from the side. To validate this, we added 0.2[Kg] to the load, and observed that the open-loop control leads to falling. At the same time, the closed-loop controller could locomote without falling, but the robot was greatly disturbed.

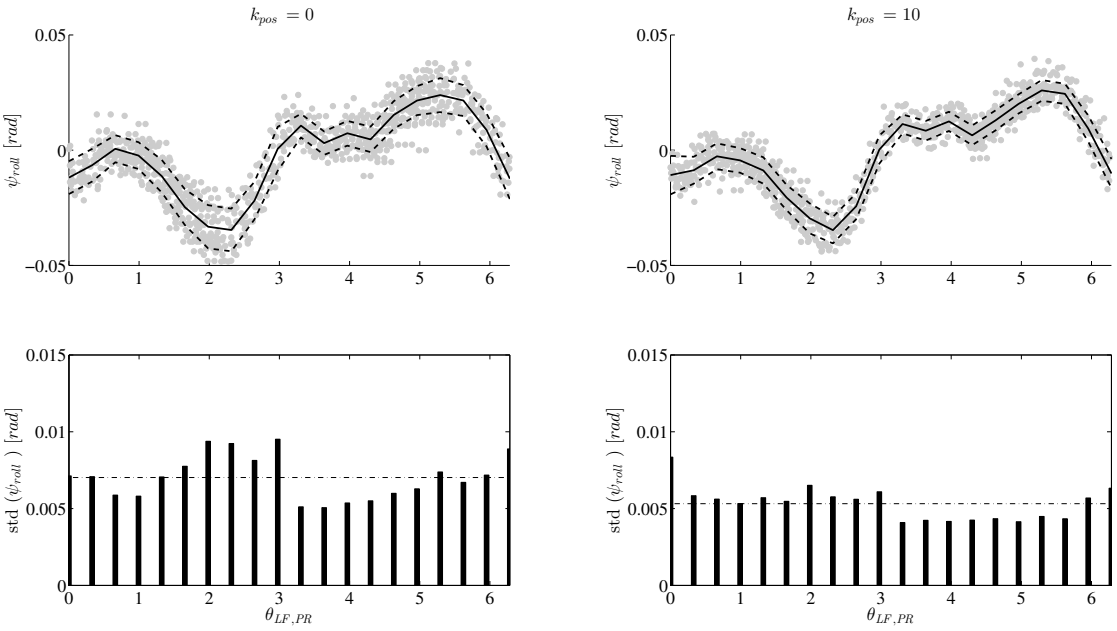


Figure 8.5: Trunk roll variations over different locomotion cycles. For the *top* plots, the gray markers are the collected data, the solid line is the mean across the cycles, and the dashed lines are showing the standard deviation over cycles. For the *bottom* plots, the bars show the values of the standard deviation, while the dashed horizontal line shows the average standard deviation over all phase values. *Left,* the open-loop controller, $k_{pos} = 0$. *Right,* the closed-loop controller, $k_{pos} = 10$, which the average roll standard deviation is 32% smaller.

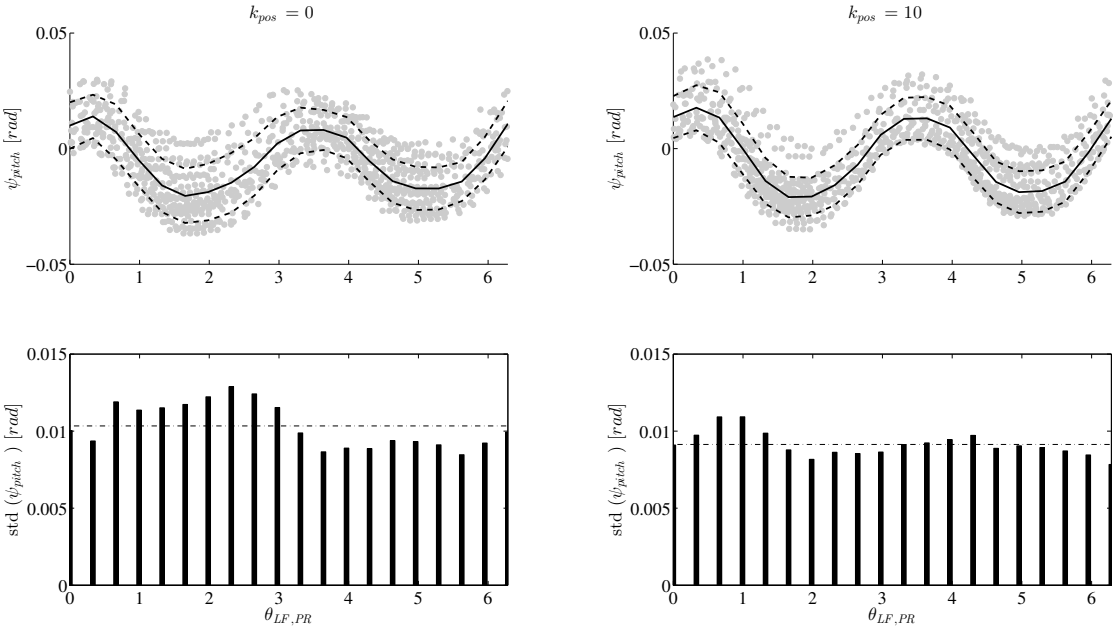


Figure 8.6: Trunk pitch variations over different locomotion cycles. Same description as in Figure 8.5. The average pitch standard deviation is 13% smaller for the closed-loop controller.

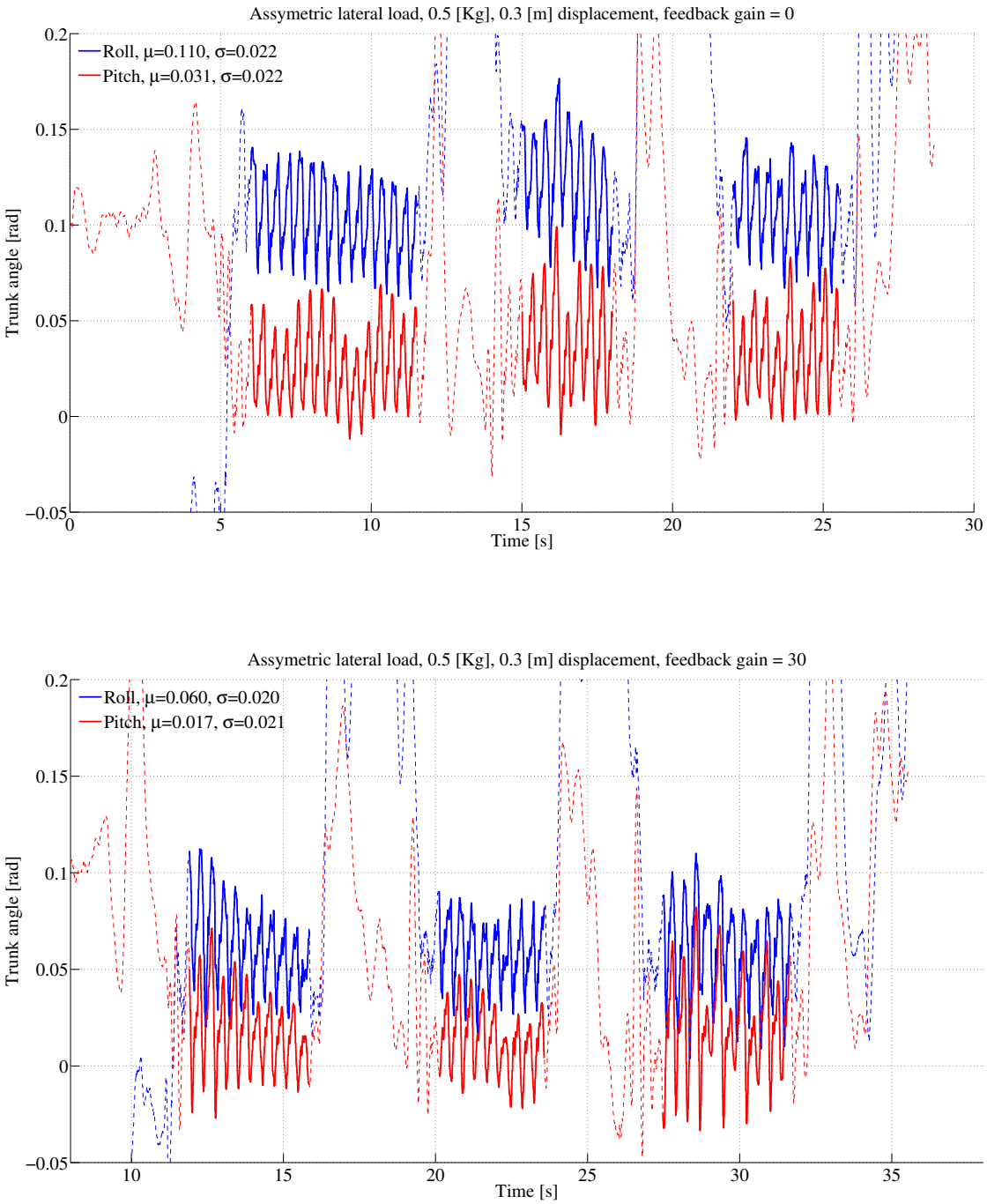


Figure 8.7: Asymmetric load carriage with and without posture control feedback. The dashed lines are for when the robot is in the air (the values can be ignored), and he solid lines are for when the robot is freely locomoting on the ground. *Top,* the feedback is turned off. *Bottom,* the feedback is on, and continuously adjusts the trunk angles, which makes the average trunk angles about two times smaller.

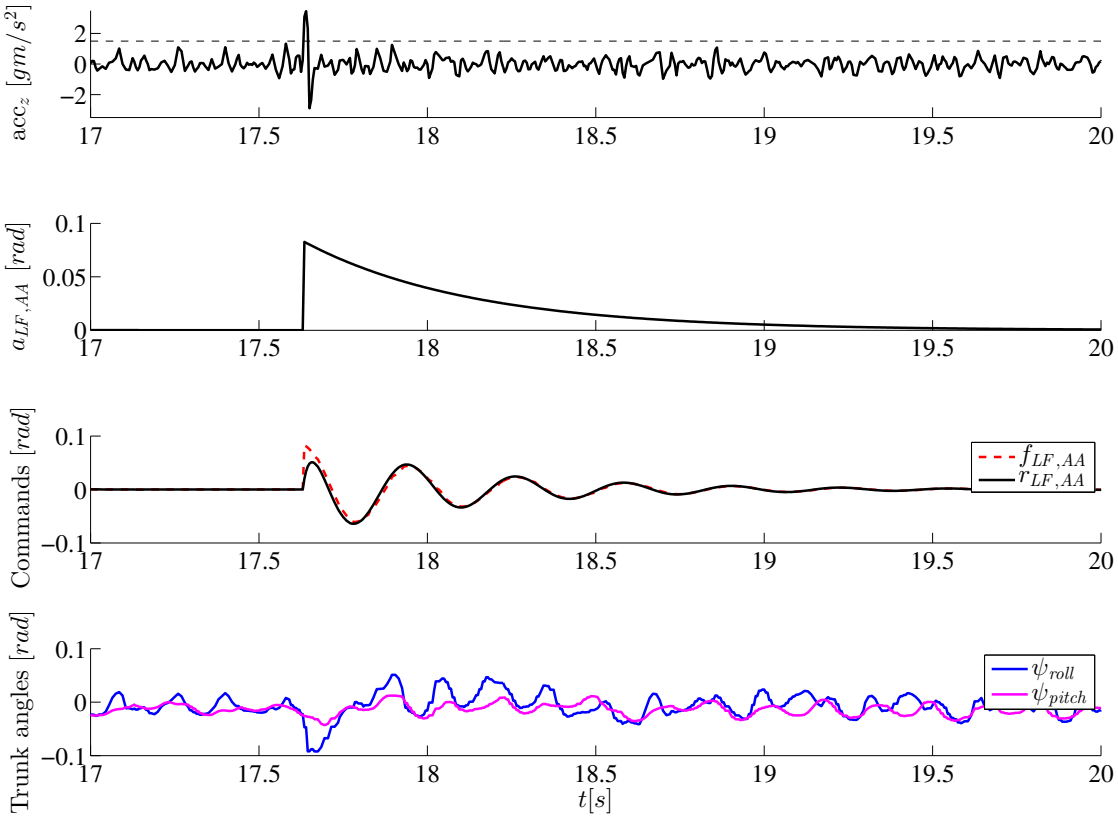


Figure 8.8: Activation of LSR after an external perturbation at about t = 17.6[s]. The robot is "slapped" with an impulse lateral forces of more than 5[N]. *First from top,* the lateral acceleration. The dashed line shows the acceleration threshold to activate the LSR. *Second,* the hip AA amplification signal $a_{LF,AA}$ is generated in response to the perturbation. *Third,* The hip AA limit cycle $f_{LF,AA}$ is discontinuously changed, but the generated motor command $r_{LF,AA}$ remains continuous and it smoothly converges to $f_{LF,PR}$. *Fourth,* The trunk roll angle values show that the robot starts to roll anticlockwise, and then recovers.

8.3.3 External Lateral Perturbations

To test the effect of the lateral stepping reflex (LSR), we "slap" the robot laterally as it walks on the flat terrain. The duration of force is rather small (impulse force), and the magnitude is typically about 5[N]. Figure 8.8 illustrates the activation of LSR in response to a lateral perturbation. The robot is perturbed at about t = 17.6[s], and the LSR is activated and amplifies the hip AA limit cycle. The robot recovers after about one stride cycle.

It is important to mention that the magnitude of the lateral perturbation cannot be increased much more than these values. This is due to the fact that the range of motion for the shoulder/hip AA joint is very small, and handling larger perturbations needs taking bigger sidesteps.

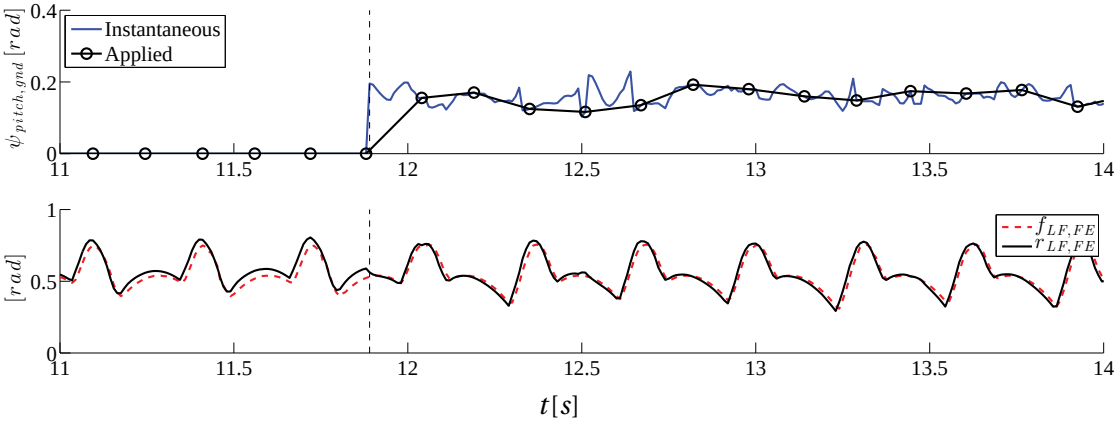


Figure 8.9: Oncilla climbing an upwards 15[deg] slope. The robot is put on the treadmill before $t = 12$[s]. *Top,* the robot estimates the ground inclination as it walks on the treadmill. The instantaneous ground inclination is estimated in all parts of the stance phase, but the update is only applied to the swinging legs in the middle of the swing phase. *Middle,* the limit cycle shape is updated as soon as the ground inclination change is sensed. The change of the limit cycle is calculated from the shift in the foot trajectory, and recalculating the inverse kinematics.

8.3.4 Inclined Surfaces

We tested the Oncilla robot with the task to climb up a 15[deg] slope (about 27% inclination). We use the foot trajectory shifting strategy to control the angle-of-attack. Using the open-loop controller, the robot always turns to one side and starts to fall from the side. With the application of the closed-loop controller, the robot travels for a considerably longer time before it starts to turn. Even when the robot turns, it does not tend to fall sideways, and can continue to climb with simple turning inputs from the operator. Figure 8.9 depicts the results when the closed-loop controller is used. The robot estimates the ground inclination as it walks on it, shifts the foot trajectory, and recalculates the joint angle limit cycles using inverse kinematics.

8.3.5 Vertical Obstacle

To test the effect of the stumbling correction reflex (SCR), we placed a vertical obstacle (about 5% of the leg length) on flat ground in the way of locomotion. Unfortunately the robot cannot always react to the obstacle quickly enough when locomotion at the typical 0.5[m/s] locomotion speed. This is mostly due to the fact that the swing leg is moving very fast (t_{sw} = 0.150[s]) and it takes at least about 0.010[s] to activate the SCR, however the robot already kicks itself backwards in this 10 milliseconds. This does not happen in all the cases, but the robot tends to stumble first, and go over the obstacle in the second (or third) try.

We reduced the locomotion speed to 0.3[m/s], increased the swing time to t_{sw} = 0.250[s], and added a halt signal to the hip PR when stumbling to prevent the robot from pushing

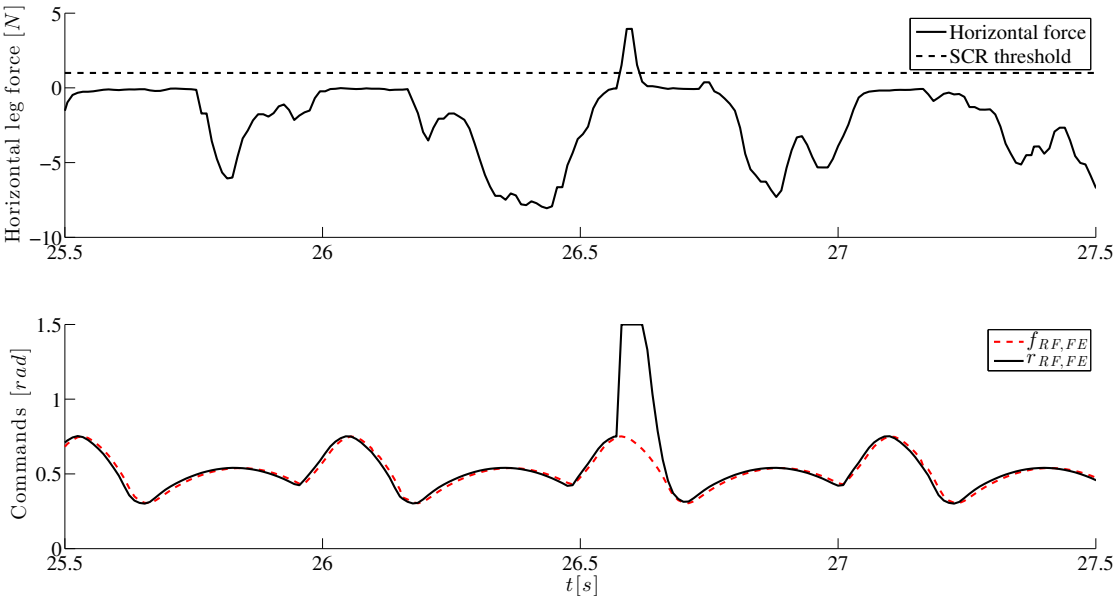


Figure 8.10: Activation of the stumbling correction reflex (SCR). *Top*, the right-fore foot hits an obstacle after $t = 26.5$[s]. *Bottom,* SCR is activated, giving feedback to the morphed oscillator node generating the reference commands. The generated command smoothly converges back to the encoded limit cycle after the foot passes over the obstacle.

itself backwards. With this setup, the robot successfully goes over the obstacle. We tested five consecutive runs with both open- and closed-loop controller. In four out of five cases, using the open-loop controller, the robot stumbled and abruptly changed direction, and in one case it passed over obstacle without stumbling. The closed-loop controller always prevented stumbling and was successful in all the five runs. Figure 8.10 shows and example of the activation of SCR. The right-fore leg hits the obstacle after $t = 26.5$[s]. This is visible in Figure 8.10-*top* which shows the horizontal force exerted on the foot. The controller activates the SCR, as shown in Figure 8.10-*bottom*. The generated command quickly and smoothly converges back to the encoded limit cycle behavior after the foot has passed over the obstacle.

8.3.6 Uneven Terrain

We built a rough terrain setup made of stairs, pebbles, and wooden pieces. The setup is depicted in Figure 8.11. The robot starts by locomoting on parquet, then climbing two small stairs, then walking on a tatami, then going down into moving pebbles, and finally walking over fixed wooden obstacles.

Locomotion on such a terrain needs the activation of SCR, and as we discussed previously, we could only activate the SCR with a slower gait. However, we realized that SCR can be activated even during full speed locomotion, if the robot trots backwards. Due to the leg design, the legs are compliant in the aft-fore direction while swinging, because of the presence of the open-pantograph spring an the cable-clutch mechanism. To clarify, the ankle joint flexes if an

Figure 8.11: Rough terrain locomotion setup used for experiments with the Oncilla robot. The robot starts by locomoting on parquet, then climbing two small stairs of 0.010 and 0.015[m], then walking on a tatami which has a considerably larger friction coefficient, then going down into moving pebbles, and finally walking over fixed wooden obstacles which can create a maximum elevation difference of 0.03[m] ($\approx$ 16% of the leg length) between the footholds.

external force is applied to the back of the foot, and the leg flexion/extension cable goes slack. This is not true for the fore-aft direction, because it needs extension of the ankle which is prevented by the cable mechanism. This directional compliance gives time to SCR to activate before the robot pushes itself backwards.

We did the rough terrain experiments with a backwards trot gait of 0.4[m/s]. The robot could trot on rough terrain with speeds as high as 0.6[m/s], however could damage itself in case of a fall. Other gains are set as described in Chapter 6. Using an open-loop controller, the robot could successfully trot from the beginning to the end of the terrain for five out of ten repetitions. In two other cases, the robot did not fall, but was very unstable and could easily fall with an additional perturbation. The closed-loop controller performed nine successful runs out of ten. For the only failure case, the robot could not prevent stumbling into an obstacle, and abruptly changed direction. This did not lead to falling, but we had to manually correct the heading direction of the robot. An example of the sensing and control signals generated during rough terrain locomotion is presented in Figure 8.12.

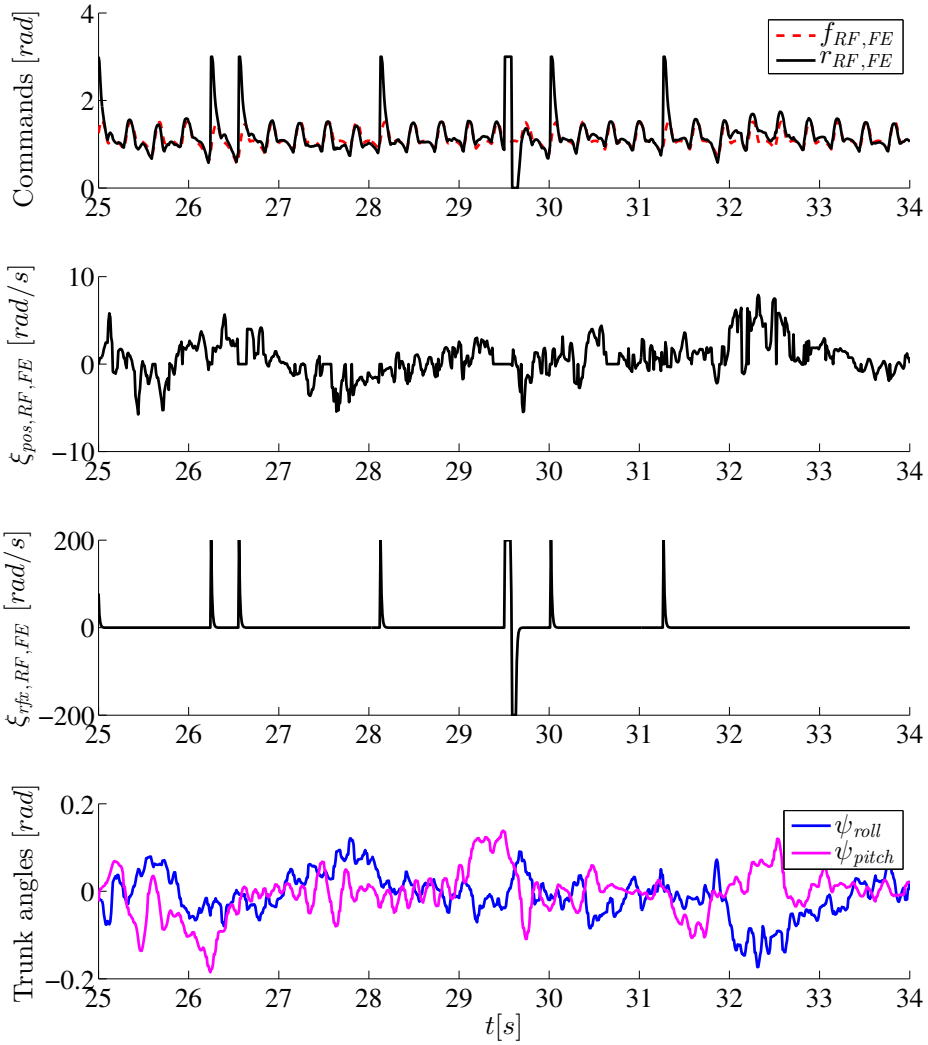


Figure 8.12: Closed-loop control on rough terrain. Control signals are for the right-fore knee FE. *First from top,* the commands generated by the CPG, which are affected by continuous and momentary feedback signals. *Second,* the posture control feedback. *Third,* SCR is activated in several cases, and in one case the LER is activated, at about $t = 29.6$[s]. *Fourth,* trunk angles during locomotion. *Overall,* the moment of climbing over the first two stairs is visible by the two consecutive SCR between $t = 26$[s] and $t = 27$[s]. Moreover, one can observe a correlation between the posture control feedback and the trunk roll, due to the fact that the trunk orientation is mostly controlled by the leg length, which is affected by the knee FE angle. Finally, after $t = 33$[s] when the robot is locomoting on flat terrain, the posture control feedback is around zero, reflexes are not activated anymore, and trunk angles are contained.

8.4 Summary

We presented the results obtained from hardware experiments with the Oncilla robot. Oncilla is a small quadrupedal robot with passively compliant pantograph legs. The robot is self-contained, the power is provided from a battery, and all the computation is done on-board at

a control rate of $100-250$[Hz]. Compared to the simulated version, the robot is about 25% heavier, and the legs are more compliant, and the range of motion for shoulder/hip AA is very limited.

The robot can locomote on flat terrain with a CPG-only control, as shown in Figure 8.3, but locomotion on rough terrain is consistently improved by closing the control loop with posture control and reflex modules. We showed that, with the application of the proposed control architecture, the robot can carry asymmetric load, react to lateral perturbations, and climb on steep inclined surfaces.

We had to reduce the locomotion speed and elongate the swing time to be able to go over vertical obstacles by the activation of the stumbling correction reflex (SCR). This is due to the fact that the legs are not passively compliant in fore-aft direction, and SCR should be activated at the very instant that the foot has hit an obstacle. This sensitivity can be alleviated by locomoting backwards, and exploiting the fact that the legs are compliant in the aft-fore direction. This compliance gives SCR an extra time for activation.

Although the robot can blindly locomote (backwards) on rough terrain with a success rate of more than 50% even with an open-loop controller, performance is improved by the application of the closed-loop control. The partial success of the open-loop controller on rough terrain comes from the passive compliance in the legs, which eliminates the need for precise foot placement. The additional performance boost from closing the loop is achieved by keeping the body attitude in check, and preventing stumbling to a considerable extent.

From our experience with the Oncilla robot, the CPG-only control is not very sensitive to the errors in robot calibration, different from the feedback modules (the posture controller and the reflex generator) which are sensitive to the calibration quality. Activation of reflexes is dependent on the state machine, and an imprecise calibration disrupts the timing of the contact phases (e.g. stance phase becomes noticeably longer for legs on one side of the robot), which can lead to untimely activation of the reflexes. Moreover, we had to reduce the posture control gain if the calibration quality was not high. This is because the robot can be slightly tilted with an imprecise calibration though walking straight, therefor the posture controller makes an effort to correct the attitude, and in turn makes the robot turn while locomotion. In conclusion, running an open-loop controller is comparatively quick and effortless, but has limited performance capacity. Closing the loop improves the performance, especially on rough terrain, if the calibration is done precisely.

9 Case Study: Sensory Feedback Delay

Life is the continuous adjustment of internal relations to external relations.

Herbert Spencer

The previous chapters discussed how a modular controller can be made by combining feedforward and feedback modules. CPGs, made of coupled morphed oscillators, construct the feedforward part of the control, while posture control and reflex modules implement the feedback part. This chapter studies how sensory feedback delay in such a system affects the performance.

The work presented in this chapter is inspired from [Kuo, 2002], which discusses the relative roles of feedforward and feedback parts in rhythmic movement control. [Kuo, 2002] utilizes a simple model of a damped pendulum tasked with sustaining steady-state oscillations of a fixed amplitude. They show that for an ideal system, both purely feedforward (CPG) or feedback controllers can be used to perform the desired task. However, the pure feedforward controller has significant performance degradation in the presence of unexpected disturbances, and the pure feedback controller is greatly affected by imperfect sensory information. They show that an optimally designed hybrid of the two can, to a good extent, handle both the external perturbations and sensory error.

In the process of designing the optimal hybrid controller, they propose that CPGs can be utilized to generate oscillatory signals to assist the decoding of sensory information instead of generating feedforward commands. For the case of [Kuo, 2002], the CPG is assumed to model the damped pendulum dynamics, and produce a prediction of motion that in turn drives the feedback part of the control.

Regarding the problem of quadrupedal locomotion, we have already discussed that for a task like blind rough terrain locomotion, which induces considerable external perturbations to the system, the feedback is necessary. However, we still need to answer why we do not use a purely feedback system, similar to e.g. [Raibert et al., 1986]. The essence of the answer is that, as showed in [Kuo, 2002], purely feedback driven systems are sensitive to sensing errors. What we present in this chapter is how sensing error *in form of sensory feedback delay* affects hybrid feedforward-feedback locomotion control.

Studies on mammals neurobiology, specifically research on nerve conduction velocity [Griggs et al., 2011], show that there are delays in the transmission of electrochemical signals traveling along neural pathways. Conduction velocity depends on an axon's diameter and how much it is myelinated, which can be different for various sensor pathways (e.g. proprioception sensory fibers have a conduction velocity of about $80-120$[m/s], while this is about $33-75$[m/s] for cutaneous mechanoreceptors). The question is how the locomotor system handles sensory feedback delay. This thesis cannot answer this question from a neuro-biological perspective, but we instead test how sensing delay affects our proposed controller to have some insight on the problem. Different from [Kuo, 2002], we keep the CPG as the feedforward command generator, and it does not take the role of processing information for a feedback controller.

9.1 Setup

We use the simulated Oncilla for the experiments in this chapter. Sensory feedback delay is introduced in the reading of vestibular (IMU) and proprioceptive sensors (joint encoders). Let us assume that the amount of delay is $\Delta\epsilon$[s]. We have:

$$n_{delay} = \Delta\epsilon / \Delta t \tag{9.1}$$

with Δt being the control timestep (typically 2[ms]). n_{delay} is the number of timesteps which represents the delay. For the experiments, we start the controllers with zero delay, and increase the delay by one timestep at each control timestep, until n_{delay} repetitions. So a delay of $\Delta\epsilon$ is applied gradually in the first $\Delta\epsilon$ seconds of the simulation. This procedure is to avoid a abrupt introduction of delay in sensory feedback in the beginning of the simulation.

We locomote on a flat terrain, and monitor the rotations of the trunk and the time to fall as measures of how delay affects the control quality. The reason why we do the experiments on flat terrain instead of rough terrain is to have clean data (which only includes the effect of delay, and not the perturbations caused by rough terrain). We assume that the behavior of the trunk angles is a good measure of how stable the locomotion controller is, as shown in the previous chapters when discussing the Role-Pitch-Variations (RPV). Therefore a controller which is demonstrating a more disturbed profile of the trunk angles is more likely to be unsuccessful on rough terrain.

We run simulations of 10[s] for each set of delay values for IMU and encoders. The same parameters as described in Chapter 7, Section 7.2 are used. Since we locomote only on a flat terrain, we change k_{pos} to the optimal value of 55 (previously illustrated in Figure 7.6). The simulation is stopped if the trunk roll or pitch exceeds 60[deg]. As the simulations are done on a flat terrain, the reflexes are not of importance, and are turned off. So the main study here is the interaction of the feedforward CPG and the feedback posture controller in presence of sensing delay. Keep in mind that delay in sensing the IMU values affects the calculation on the rotation matrix describing the trunk orientation, and encoder delay affects the correct calculation of the Jacobian matrices used by the posture controller.

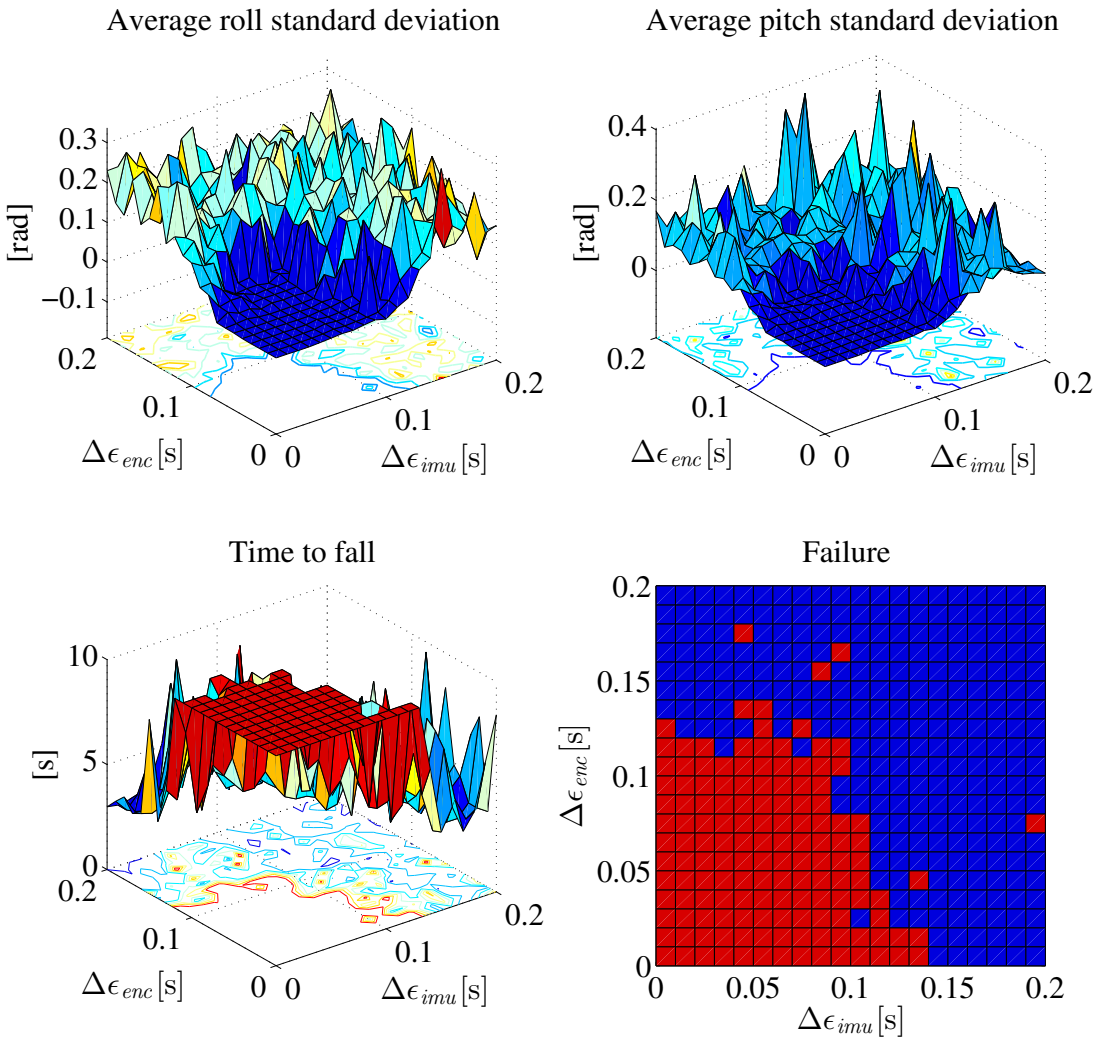


Figure 9.1: The effect of IMU and encoder delays on the performance. For these experiments $k_{pos} = 55$. Delays are in seconds. *Top-left* and *top-right* show the average roll and pitch standard deviations for different combinations of delay values. Locomotion with IMU and encoder delays in $[0, 0.09] \times [0, 0.11]$ does not lead to a fall within the 10[s] of simulation time, as shown in *bottom-left* and *bottom-right*.

9.2 Results

Before we present the results of perturbing the control by sensing delay, we would like to mention that we intuitively expect the performance to decay as the delay is increased. Figure 9.1 illustrates the results when IMU and encoder delays are increased up to 0.2 [s], which is 50% of the typical stride duration. As measures of how good the controller is working, we stack the roll and pitch values over different cycles, calculate the standard deviation for each phase value, and then average the standard deviation values. Figure 9.1-*top-left* and -*top-right* illustrate these average standard deviation values for different combinations of IMU and encoder delays. Figure 9.1-*bottom-left* shows the time to fall, and the *bottom-right* plot

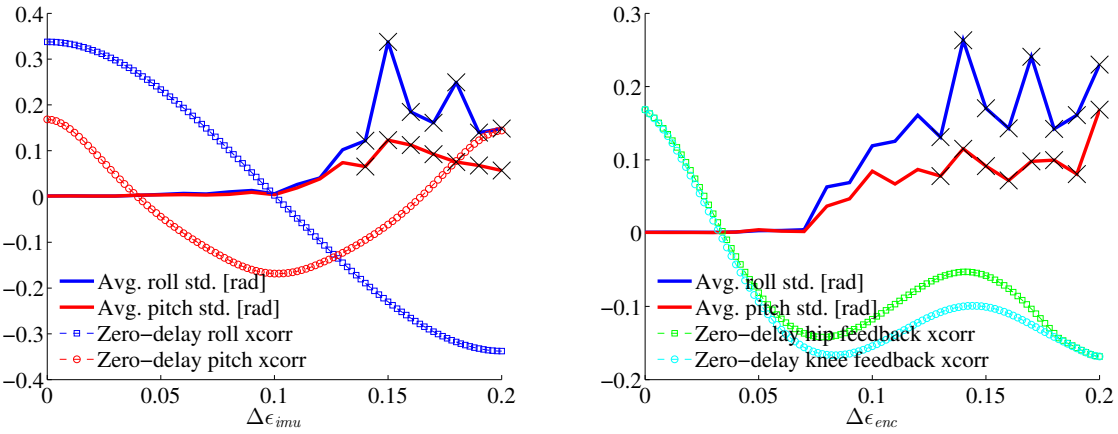


Figure 9.2: The effect of delay on performance for single delays and the indicators. For this figure, $k_{pos} = 55$. Black crosses show the cases where the robot falls within the 10[s] of simulation. *Left,* the effect of IMU delay. The zero-crossing of roll auto-correlation (xcorr) is a good indicator of the effect of IMU delay. *Right,* the effect of encoder delay. The zero-crossing of feedback auto-correlation is a good indicator of the effect of encoder delay. Auto-correction values are normalized to the plot range.

shows if the robot falls within the 10[s] of simulation.

What is somehow counterintuitive is that, instead of gradually affecting locomotion, delays do (almost) not affect locomotion up to some threshold (around 0.05[s]). Only when above this threshold does the effect of delay become visible. One can take a look at the videos provided online along this thesis, and compare the locomotion when delay is zero and when it is about 0.03[s], which does not show any visible difference. This is interesting, as it shows that if the delay in the locomotor system is bounded, and the locomotor system is a hybrid feedforward-feedback system, then perhaps it does not affect the performance of the system. Of course the analysis here is at a very high level of abstraction compared to a neurobiological system.

We would like to find indicators which predict what amounts of delay deteriorate the performance. Take the example of attitude correction. One hypothesis is that as long as the delay does not make the correction of the posture anti-phase compared to a zero-delay correction, then it should not affect the control in a destructive manner.

For the case of IMU delay, we calculate the auto-correlation (periodic self cross-correlation) of the roll and pitch profiles, for a zero-delay controller. Figure 9.2-*left* depicts the auto-correlation profiles, and the performance measures when IMU values are delayed. We can see that the zero-crossing of the auto-correlation profiles are good candidates to predict if the system is going to be greatly affected by the delay. If we zoom-in on the profiles, as shown in Figure 9.3, we see that the roll auto-correlation is a better indicator than the pitch auto-correlation. This is due to the fact that the quadrupeds typically have a bigger shoulder to hip separation compared to lateral shoulder (or hip) separation, which makes the balance more

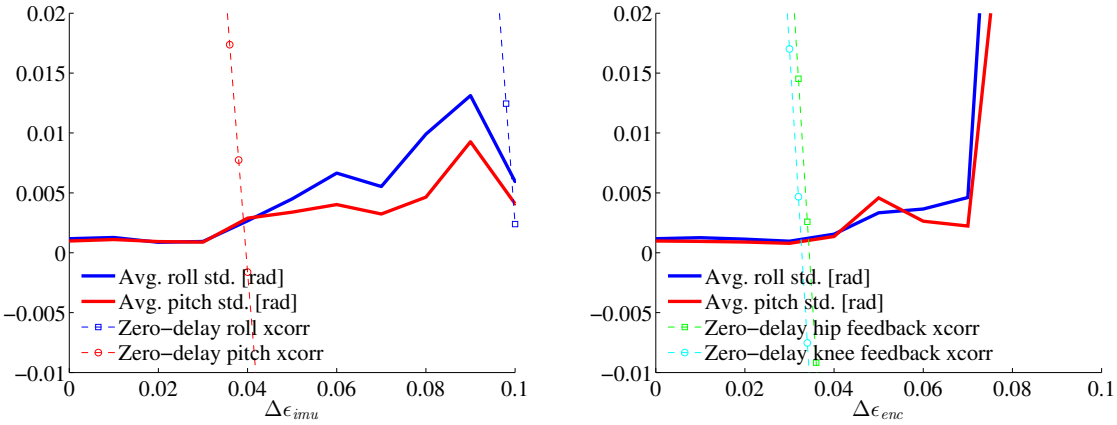


Figure 9.3: Zoom-in of Figure 9.2. *Left*, we see that the roll auto-correlation is a better indicator of the IMU delay effect compared to the pitch auto-correlation. *Right,* both hip and knee posture control feedback auto-correlation are good indicators of the encoder delay effect.

sensitive to roll variations. We will see later when presenting the additional experiments that roll auto-correlation is always a good indicator of whether the IMU delay will greatly perturb the system. This is not necessarily true for the pitch auto-correlation.

For the case of delay in sensing the encoder values, the results are presented in Figure 9.2-*right*. The encoder values are only used to calculate the leg Jacobian matrices, and these Jacobian matrices are used to calculate the posture control feedback $\boldsymbol{\xi}_{pos}$. So as a possible indicator, we calculate the auto-correlation of the hip and knee posture control feedback signals for one of the legs, left-fore in this case ($\xi_{pos,LF,PR}$, $\xi_{pos,LF,FE}$), for when there is no delay. As Figure 9.2-*left* shows, the zero-crossings of the feedback auto-correlation signals are good indicators to determine if the encoder delay is going to affect the balance. To explain in a simpler way, if the zero-delay correction for a knee is to momentarily flex, then the amount of delay which makes the knee extend instead of flexing is destructive.

9.2.1 Additional Test: Stronger Feedback

We showed that the feedback part of the system is affected by the sensing delay, but only after a specific amount of delay is introduced. The results presented previously are for the case when the posture control gain was chosen to be *optimal* (refer to Figure 7.6), $k_{pos} = 55$. We now test how delay affects the system when the posture controller is with a higher gain, as chosen for the rough terrain experiments, $k_{pos} = 125$. Figure 9.4 illustrates the results for when $k_{pos} = 125$. As we can see, the system is now more sensitive to the feedback delay. This is more pronounced on the IMU delay (Figure 9.5), as it directly affects the generation of the feedback signals, while the encoder delay only affects the posture control feedback indirectly through the Jacobian matrices. We still observe that the zero-crossing of the roll auto-correlation is a good indicator for the effect of IMU delay, and the zero-crossing of the posture control feedback auto-correlation is a good indicator for the effect of the encoder delay.

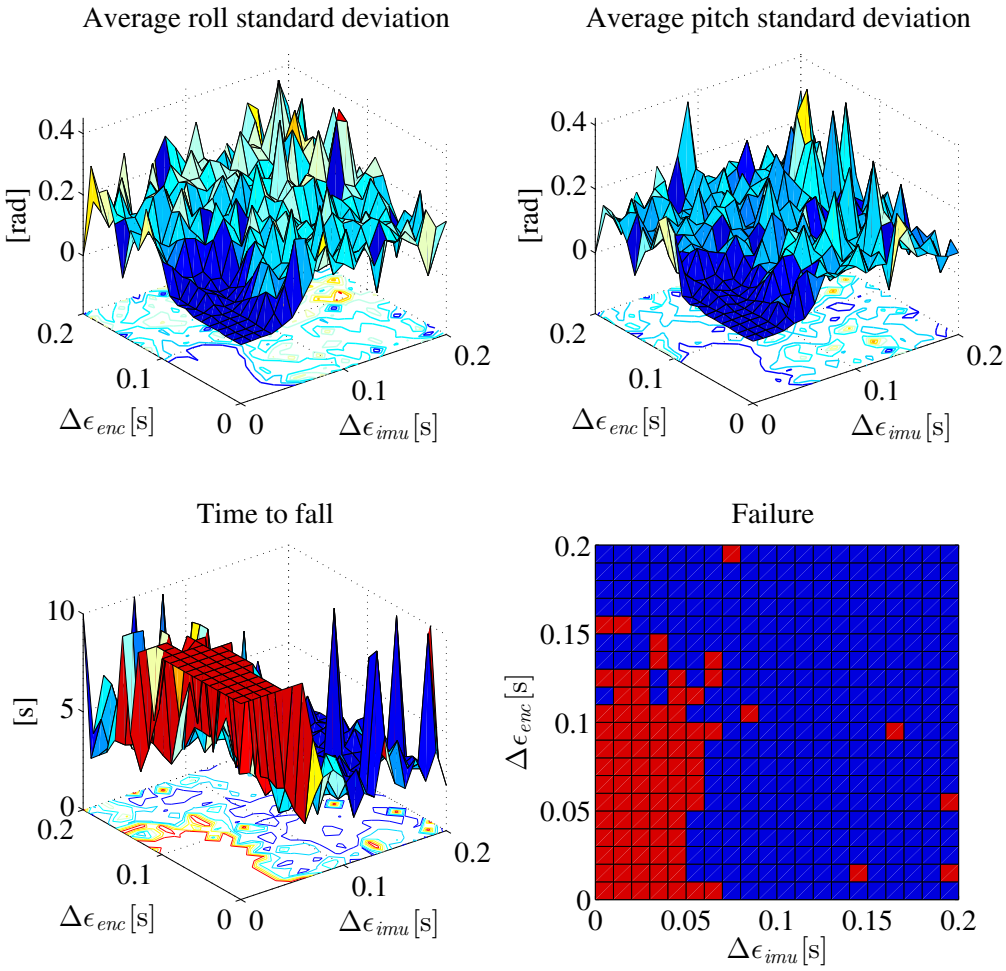


Figure 9.4: The effect of IMU and encoder delays on the performance when $k_{pos} = 125$. Annotation is as same as in Figure 9.1.

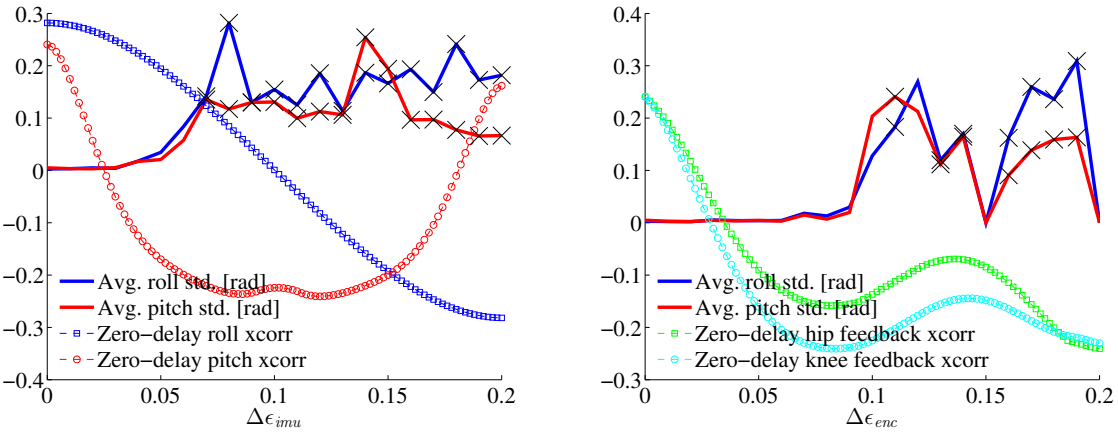


Figure 9.5: Indicators for the effect of delay when $k_{pos} = 125$. Annotation is as same as in Figure 9.2.

9.2.2 Additional Test: Different Compliance

We additionally test the effect of delay when the leg diagonal springs are 10% softer or harder compared to the default case (that has been optimized for the Oncilla robot). We use the same parameters used for the original delay experiments (same CPG, same posture controller with $k_{pos} = 55$). As illustrated in Figures 9.6-9.9, we see that a change of compliance makes the system slightly more sensitive to the effect of delay. This can be explained by the fact that a change in compliance needs a re-tuning of the posture control gain, which has been omitted here.

9.3 Summary

Inspired from [Kuo, 2002], we performed experiments to assess the effect of imperfect sensory information, in form of time delay, on the quality of locomotion. [Kuo, 2002] states that a purely feedforward controller is sensitive to external disturbances, and a purely feedback controller is affected by the sensory noise. Thus, the optimal solution is to have a hybrid feedforward-feedback controller.

Our experiments show that, though unintuitive, the sensory delay does not greatly affect the system if the delay is bounded to less than about 10% of the stride duration (the zero-crossing of the roll auto-correlation in Figure 9.3). There seems to be a bifurcation in the coupled dynamics of controller and forward-dynamics physics after this bound. We only show this bifurcation behavior empirically, and performing a theoretical analysis can be a line of research for future.

We proposed that, for vestibular delay (IMU), the auto-correlation of the roll values of an unperturbed controller is a good indicator of the delay effect. The zero-crossing of the roll auto-correlation profile is the point where the delay starts to be negatively effective. As for the proprioceptive delay (encoders), the auto-correlation of the posture control feedback signals of an unperturbed controller is the indicator of the delay effect. The sign inversion of these feedback signals, caused by delay, is the starting point of being sensitive to delay.

The results presented here are based on a locomotion controller which has been designed with a high-level of abstraction compared to a detailed neuro-biological model. Nevertheless, the results give some insights on how the neuro-biological system might be able to handle sensing delay in the neural pathways, assuming that the locomotor system is a hybrid feedforward-feedback system. One hypothesis can be that the presence of CPGs might have been the local-optimal solution to deal with delays as the neural systems have evolved, though this needs an in-depth study which is not within the scope of this thesis.

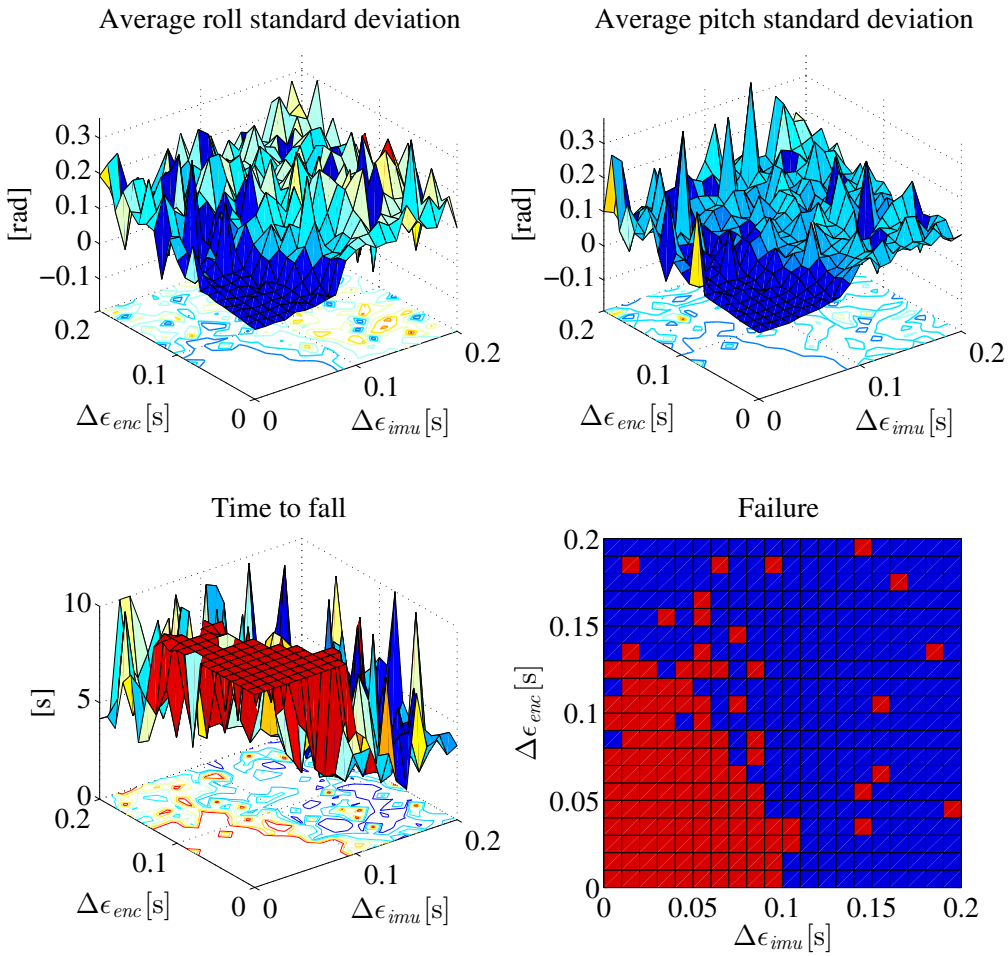


Figure 9.6: The effect of IMU and encoder delays on the performance with 10% softer leg springs. Annotation is as same as in Figure 9.1.

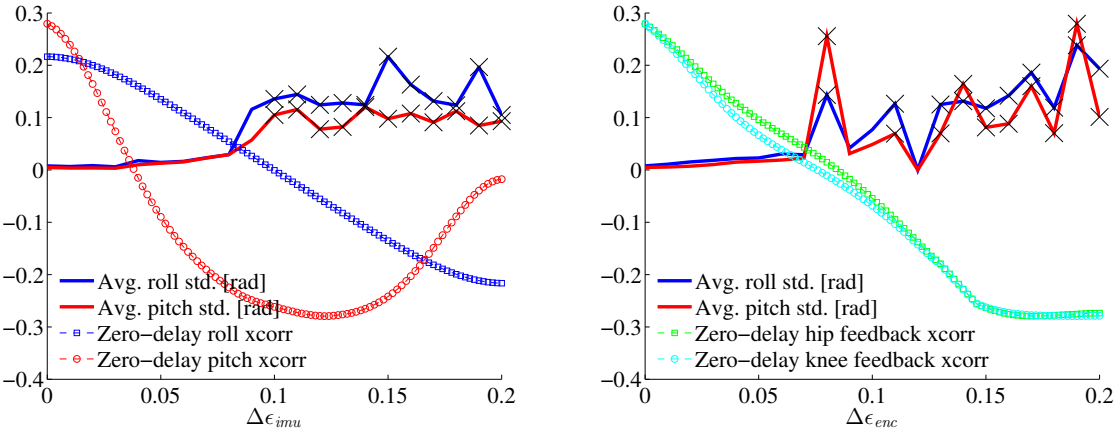


Figure 9.7: Indicators for the effect of delay with 10% softer leg springs. Annotation is as same as in Figure 9.2.

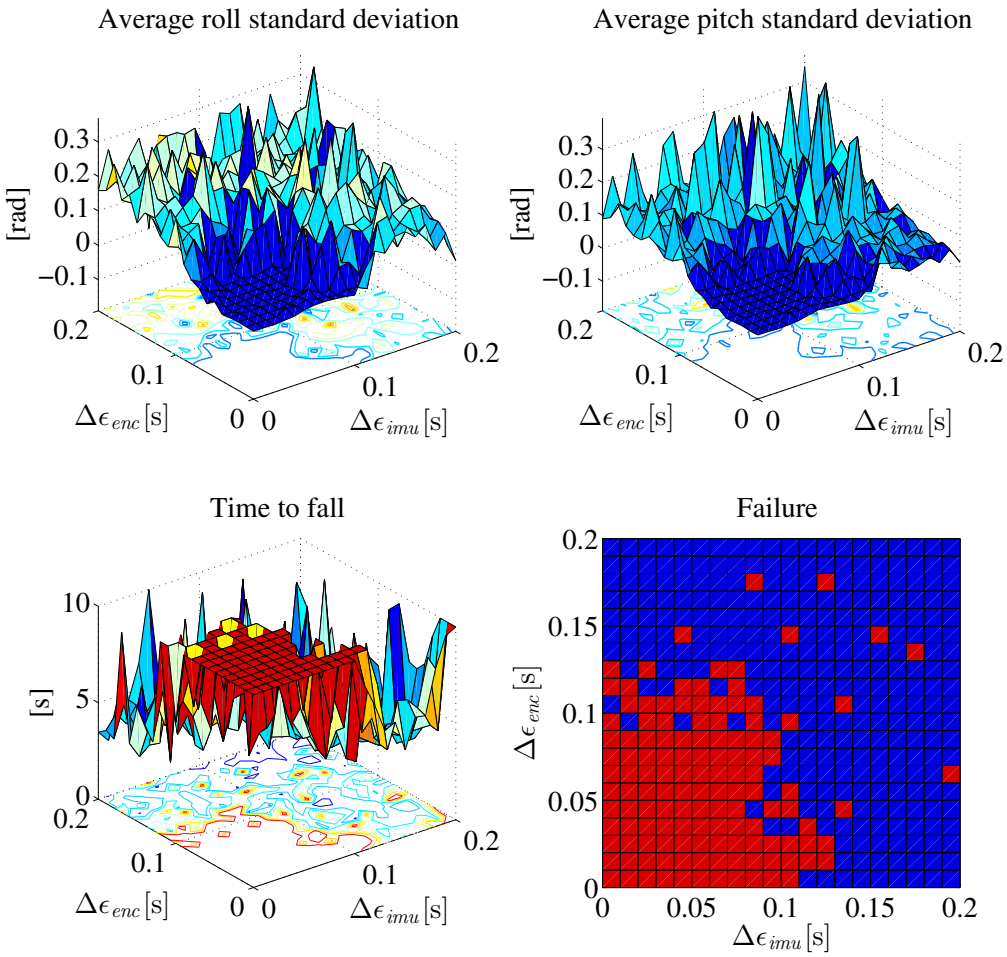


Figure 9.8: The effect of IMU and encoder delays on the performance with 10% harder leg springs. Annotation is as same as in Figure 9.1.

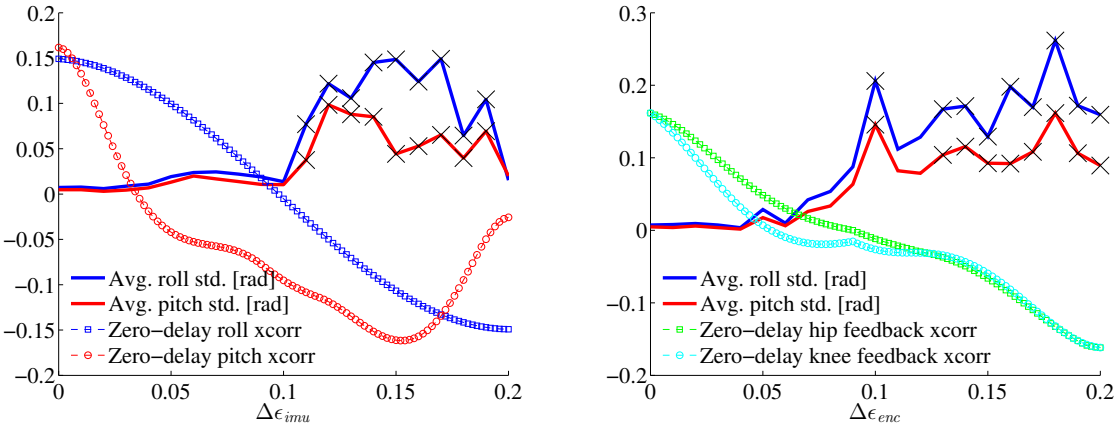


Figure 9.9: Indicators for the effect of delay with 10% harder leg springs. Annotation is as same as in Figure 9.2.

Conclusion Part IV

10 Discussion

Animals locomote on rough terrain seemingly without any difficulty. This does not mean that the locomotor system behind this is simple. Conversely, the locomotor system is quite redundant and complex at different levels, and locomotion is the result of the interaction of many components. Moreover, the complexity of the locomotor system is increased by the closing of the control loop with sensory feedback to different levels. Here in this thesis we studied how a few control modules at a high level of abstraction can construct a controller for locomotion with quadrupedal robots.

We chose the problem of *blind*, as opposed to visually-guided, rough terrain locomotion as the target of experiments. Blind rough terrain locomotion requires continuous and momentary corrections of leg movements and body posture, and provides a proper testbed to observe the interaction of different modules involved in locomotion control. As for the specific case of this thesis, we had to design rough terrain locomotion controllers which could benefit from torque control but do not depend on it, have limited sensing (e.g. no 3D contact force sensing), and have to be computationally light, all due to the properties of the robotics platform that we use.

We proposed that a robust locomotion controller, taking into account the aforementioned constraints, is constructed from at least three modules: 1) pattern generators providing the nominal patterns of locomotion; 2) posture controller continuously adjusting the attitude of the body and keeping the robot upright; and 3) quick reflexes to react to unwanted momentary events like stumbling or an external force impulse.

A suitable pattern generator should be able to generate smooth output even in presence of discontinuous feedback, be capable of encoding desired limit cycle behaviors, and be simple to construct and use. We introduced morphed oscillators that inherit all the aforementioned properties. Morphed oscillators are constructed by morphing the limit cycle of a known oscillator which has desired asymptotic stability properties through a phase-dependent shaping function. We showed that the framework of morphed oscillators can be used to design a whole family of nonlinear oscillators with arbitrary limit cycle shapes and traceable stability properties. Moreover, their output can be modulated on-the-fly without creating discontinuities, including the shape of the output signal.

The framework of morphed oscillators systematizes the process of oscillator design and eases the process of creating CPGs in form of coupled oscillators. If a CPG constructed from coupled morphed oscillators is set up correctly by a proper choice of the limit cycle shaping functions, then it leads to an open-loop stable locomotion controller which works acceptably on flat terrain. However, to be able to blindly locomote on rough terrain, posture control and reflex mechanisms are needed. To cultivate our initial ideas, we borrowed concepts from virtual model control (VMC) [Pratt et al., 2001], and designed a posture controller module to keep the body upright while locomoting (only simulation). However, VMC generates torques and as our hardware robots do not have torque-control capability, we had to modify the posture controller. We suggested that the posture control task can still be fulfilled if one assumes that the virtual forces generated by VMC are virtual velocities, and generate joint velocity feedback signals by utilizing the Jacobian inverse method. Interestingly, because CPG is generating joint velocities, posture control feedback can be directly introduced within the CPG dynamics.

The posture controller keeps the body upright during locomotion, but reflexes are needed to react to momentary events. Inspired from what we observed in cats, guinea fowls, and successful rough terrain robots, we implemented simple impulse-based feedback mechanisms to flex the leg if the robot is stumbling (stumbling correction reflex), extend the leg if an expected contact is missing (leg extension reflex), or to initiate a lateral stepping sequence in response to a lateral external perturbation.

CPG, posture controller, and reflex generator, along with additional modules, were put together in a modular control architecture. We introduced a gait controller that transmits operator's instructions to the other modules by updating stride duration, gait type, etc. A foot trajectory regulator is used to update foot trajectories in response to a speed change or turning request, or if the foot trajectories should be shifted forward or backward to help locomote on inclined surfaces. We used a slope estimator to calculate the local inclination of the ground on-the-fly as the robot walks on different parts of the terrain. Finally, a state machine was utilized to control the timing and activation of the posture controller and reflexes. In the whole process of building up the control architecture, we explained how each module is set up and tuned, to ease the process of reimplementing the control architecture for other interested researchers.

We used two simulated quadrupeds to assess the performance of our proposed control architecture in blind rough terrain locomotion scenarios. The first quadruped, Ghostcat, is a stiff-by-nature cat-sized robot. For the experiments with Ghostcat we assumed that torque-control capability is available, and used it to test our posture control ideas inspired from VMC. The closed-loop controller (CPG + posture controller + reflex generator) helped Ghostcat blindly locomote over terrains for which a CPG-only controller was failing. The scenarios included locomotion on rocky setups, randomized uneven terrains, or inclined surfaces, and handling external forces. We also showed that the controller is robust to changes of the tuned parameters or the used gait, and could still perform acceptably with slightly varied control gains or nominal gaits. Moreover, we showed that closing the loop with the posture controller can be exploited to correct deficient open-loop controllers, or improve the working ones, by

making the role-pitch-variations (RPV) of the robot more periodic and consistent.

The second simulated quadruped which was used for experiments is Oncilla, which is modeled after its hardware counterpart. Different from Ghostcat, Oncilla legs are passively compliant and there is no torque-control capability. We did similar rough terrain locomotion experiments with Oncilla, which were concluded with similar results: a CPG-only controller is not enough, and the whole closed-loop controller provides the capability to blindly locomote on rough terrain. We additionally showed that if the closed-loop controller is utilized, then the robot is capable of carrying asymmetric loads. We illustrated that RPV of an open-loop controller while carrying asymmetric load is very noisy and disturbed, and RPV becomes periodic by closing the loop. Finally, we demonstrated that the proposed control architecture performs precise speed control by controlling the step length.

Hardware experiments with the Oncilla robot were performed to validate the results obtained in simulation. Oncilla is a self-contained small quadrupedal robot with passively compliant pantograph legs. Compared to the simulated version, the robot is about 25% heavier, and the legs are more compliant, and the range of motion for shoulder/hip AA is very limited. Taking into account the limitations of the hardware, we had to reduce the difficulty of the terrains used for hardware experiments (compared to the simulation setups). Nevertheless, we showed that by applying the control architecture proposed in this thesis, the robot can safely carry asymmetric load, handle external perturbations, locomote on inclined surfaces, and blindly traverse over uneven terrain. We found out that locomoting backwards gives an advantage on rough terrain, due to the leg design of the Oncilla robot which creates directional compliance.

Additional to using the proposed control architecture to demonstrate successful blind rough terrain locomotion, we decided to use it to study, at a high level of abstraction, the relation of feedforward and feedback modules in the locomotor system. Thus we performed experiments to assess the effect of the proprioceptive and vestibular sensing delays on the quality of locomotion. One can think that the expected effect of delay is to bring a constant deterioration of performance as delay increases. Counterintuitively, our experiments showed that sensing delay does not greatly affect the system, up to a bounded value, and starts to visibly lower the performance after that. To explain this, we proposed that the bifurcation point related to the vestibular delay is when the delay starts to cause the roll correction to have an opposite sign than expected. As for the proprioceptive delay, the bifurcation point seems to be the amount of delay which makes the posture control feedback have an opposite sign compared to a zero-delay controller.

To summarize, the contributions of this thesis are:

- We introduced a modular control architecture for blind rough terrain locomotion with small robots, when ingredients like torque-control capability or 3D force sensing are not available;
- We presented a systematic way to design nonlinear oscillators with arbitrary limit cycle shapes, known as morphed oscillators;

- We integrated the concept of Central Pattern Generators (CPGs) with kinematic model-based posture control;
- We showed that incorporating the posture control feedback within CPG dynamics can lead to correction of the overall behavior of the whole system (locomotion controller + forward dynamics physics);
- At a high-level of abstraction, we showed how sensing delay affects a hybrid feedforward-feedback locomotor system. We presented empirical results suggesting that delay does not greatly affect the system up to a bifurcation point.

The rest of the material presented in this Chapter is organized in two sections. We first qualitatively compare the proposed locomotion control framework to other existing methods. After that, we provide a discussion in form of questions and answers that covers different aspects of this thesis.

10.1 Comparison

The locomotion control framework presented in this thesis only covers the high-level part, and does not cover the low-level motor control part. Thus, we only address the high-level pattern generation part of the locomotion control presented in the state of the art.

The closed-loop pattern generation framework presented in this thesis has several similarities to [Fukuoka et al., 2003, Kimura et al., 2007b]. As detailed in Chapter 2 about the Tekken robot, [Fukuoka et al., 2003, Kimura et al., 2007b] utilize Matsuoka oscillators as the pattern generator, and include a number of reflex/reaction modules providing feedback. Though their results are promising, the control methodology is rather complicated. Tuning Matsuoka oscillators to generate desired rhythms needs human intuition or numerical optimization, and even after that the basin of attraction is bounded. We introduced the morphed oscillators that take the role of pattern generation while easing the design process. Any desired limit cycle shape can be explicitly encoded into morphed oscillators without any optimization, and the basin of attraction can be infinitely large, and analytically known. As for the posture control, [Fukuoka et al., 2003, Kimura et al., 2007b] use reaction behaviors to adjust the posture whose effectiveness is based on observation, and trial and error. Here in this thesis, we formulated the posture control problem as a kinematic model-based feedback generation procedure, and only left a few (sometimes only one) gain parameters to be tuned. Finally, the way we introduced reflexes is similar to [Fukuoka et al., 2003, Kimura et al., 2007b], but we keep the reflexes as simple as possible (decayed impulse signals) to reduce the complexity of the hand-made parts of the control.

Another successful approach to locomotion control is the one presented in [Barasuol et al., 2013], also presented in Chapter 2 when discussing HyQ. They use a pattern generator in task-space instead of joint-space, and then do the attitude control in the null-space of the feet positions. Thus, theoretically, the attitude control does not affect the foot placement procedure. The first question to answer here is why we implement the pattern generator at the

joint-level. Our answer is, because we have developed the morphed oscillators, we can encode any arbitrary limit cycle into the joint-level pattern generators, and as we showed in Chapter 6, task-space foot trajectories are also an option. The second question is why we do not perform the posture control in the null-space of the nominal pattern generation. The answer to this question is twofold:

1. If the nominal pattern generation is optimal, i.e. it provides a gait on the flat terrain with the optimal roll and pitch variations (w.r.t. a chosen measure), then the attitude control should be performed in the null-space of the pattern generator in order to avoid disturbing the gait. However, at least for the case of our experiments, the open-loop controller is not providing an optimal RPV, as shown in Figures 7.2 and 7.6. The implementation of the posture control in this thesis is additive to the CPG dynamics, which means that they compete at the pattern generation level. Nevertheless, we showed that such a competition at the pattern generation level can lead to RPV correction.
2. Our way of implementing the posture controller does not lead to locomotion with zero roll and pitch, and instead it cleans-up the RPV. However if the reason to have an attitude controller is to have zero roll and pitch (e.g. for careful load carrying), the null-space posture controller is a better choice.

The last approach that we compare our control framework to is Raibert's control, extended to BigDog and LS3 [Raibert et al., 1986, Raibert, 2010]. The main components of the Raibert's control is to support the body by vertical hopping, adjust the body attitude by modifying the hip torques in the stance phase, and exploit foot placement to control the speed and the kinetic energy of the system. For our small robots, specially Oncilla, the robot is not able to do a hopping motion as the extension of the legs are passive through the application of springs, and the springs are not strong enough to make the robot jump. Moreover, Oncilla does not have the torque-control capability to exploit that for attitude control. Finally, Raibert's control depends on a precise estimation of the trunk velocities, which needs a state estimator, and it has been omitted from our control framework because of computational complexity. One can say that the Raibert's method controls the robot by taking actions which directly affect the dynamics of the system (e.g. kinetic energy), and our proposed control framework does dynamic locomotion control through kinematic manipulations. In an ideal case, a controller that takes into account the dynamic properties of the system is superior because the interaction with the environment is through the application of forces, but it needs additional sensors and more capable motor controllers, which are not always available/affordable for small robots.

10.2 Questions

Is the whole framework of morphed oscillators needed for locomotion control?

The answer is yes and no. As we stated in Chapter 6, only a simple version of the morphed oscillators, which uses an amplitude controlled oscillator as base, is used in this thesis for locomotion control. One can implement the behavior of such an oscillator with feedforward phase-based signal generation plus additional if-then rules and smoothing procedures to converge to a steady-state desired output, which can be time-varying (e.g. change of the limit cycle shape based on the feedback). However, extra attention should be paid to relieve such an implementation from pattern generation delay (example: a simple PD controller trying to generate a time-varying output has delays). The morphed oscillator theory provides a simple, elegant and concrete mathematical framework to construct locomotion pattern generators and trace the behavior of them. Moreover, morphed oscillators provide means to define the convergence behavior which provides several possibilities like the ability to define different transient dynamics for swing and stance phases, and this can be a line for future research.

Nevertheless, we developed the morphed oscillator not only for locomotion control, but as a generic tool to design custom-made phase oscillators. We introduced possibilities like arbitrary convergence behavior, or having an nth order desired oscillator, which are not used in this thesis, but have theoretic abstract mathematics value.

In this thesis no phase modification in reaction to the rough terrain is considered. Why?

The design of morphed oscillators allows for having feedback on both phase and output (radius), see Equation 5.2. Feedback on radius dynamics $\dot{r}_i$ directly affects the output, and this can be exploited to directly alter the generated trajectory in a traceable way. However, feedback on the phase dynamics $\dot{\theta}_i$ will first affect the phase, and then indirectly affect the output since $\dot{r}_i$ depends on $f_i(\theta_i)$. Hence, if one desires to affect the generated output spatially, it makes sense to have the feedback on radius dynamics, and this has mostly been the case for this thesis. Nevertheless, feedback on phase is important when direct temporal adjustments are needed. A good example is presented in [Owaki et al., 2013] which shows that coupling between the legs can be purely physical and based on contact force feedback. As this thesis has left the slot for phase feedback empty, it is possible to easily incorporate this concept to our proposed controller, and this is a direction for future research.

Is the proposed control architecture the best solution for blind rough terrain locomotion?

Considering the constraints that we have stated in this thesis, the proposed control architecture is a satisfactory option. It is easy to implement, and has a low computational complexity. However, the proposed control architecture does not extensively address the dynamic properties of a fast moving multibody system, like the trajectories of the center of mass. We believe

that inclusion of such information in control is rather crucial to locomote on rougher terrains compared to what is presented in this thesis.

It is important to note that rough terrain locomotion is a complex problem full of interactions with the environment, and the interactions are happening through forces and torques. Without the possibility to do force/torque control, the ways to implement a successful locomotion controller are rather limited.

Are small robots like Oncilla proper tools to study rough terrain locomotion?

Experimenting on a small robot, like Oncilla, comes with pros and cons. The cons are as stated before: limitation on sensor integration and control modes, and computational power limitations. One key factor creating these limitations is the overall weight of an autonomous robot with respect to its size (Oncilla is already overweight). The pros in using a small robot are the ease to perform experiments and the low cost of constructing and maintaining such a robot. Most of the experiments in this thesis are done with one person handling the robot, only for the cases where something might go wrong (e.g. robot hitting a wall). Moreover, the danger in using a robot such as Oncilla is very small compared to bigger powerful robots (e.g. HyQ, BigDog, etc). Overall, a small robot is a suitable tool to test hypotheses about locomotion, but is far from being applicable to real world robotics problem such as search-and-rescue.

What should be improved with the Oncilla hardware to improve the overall performance?

The Oncilla robot is quite robust, and does not need regular maintenance. However several upgrades can be considered:

- The robot is rather heavy, but it is not an easy task to reduce the weight of the robot. lighter motor-control boards might be one way to reduce the weight of the robot. The motor-control boards used on Oncilla are made in-house, and we believe that the size and weight can be reduced if the boards are made by a specialist company;
- The range of hip abduction/adduction is very limited. Lateral foot placement is important for rough terrain locomotion, and this needs a larger range of motion in the lateral degrees of freedom;
- The calibration procedure is not very precise and markedly affects the performance. This arises from the fact that hip AA joints are not equipped with encoders. Also, the knee and ankle FE joints are actuated by a proximal motor through a cable mechanism, so the calibration should be done on the motor axis, but with respect to the knee and ankle readings;
- The maximum foot clearance of the hind legs can be improved. Oncilla has longer hind legs, which means that they need to flex more in order to follow a foot trajectory similar to the fore legs. Together with the fact that the robot is heavy and crouched, this makes the normal stepping of the hind legs to almost reach the maximum foot clearance.

10.3 Future Directions

Morphed oscillators. We developed the concept of morphed oscillators and applied them to the problem of pattern generator design. We did not harness all the capabilities of the morphed oscillators for the specific problem of rough terrain locomotion. Morphed oscillators can be designed at a desired dynamical system order, which can bring properties like acceleration or jerk continuity. Moreover, they can encode arbitrary convergence behavior, which is not exploited in this thesis. We are actively searching for applications in system design and control which can benefit from the framework of the morphed oscillators and exploit their features.

Posture control integration. We showed that the posture control mechanism can provide feedback for the CPG, or adjustment torques for the motor controller. One experiment that we would like to perform is to use a torque-controlled quadruped and activate the posture controller at both levels simultaneously. It would be interesting to see how the two posture control pathways should share the task to obtain an optimal solution, and which one should have a relatively higher gain.

Smart low-level control. One question that we would like to answer in the future is "what would be the benefit of adding a smart low-level controller like inverse dynamics to our framework?". We started to perform experiments on StarlETH, in Autonomous Systems Lab, ETHZ, and are trying to make our control framework work on it. If this step succeeds, then we are interested to implement an inverse dynamic controller and observe the difference when it is tuned on or off. We expect that the advantage of the inverse dynamics controller will be revealed in cases like stumbling. With a position controller, stumbling to an obstacle would exert considerable forces to the robot until the presence of the obstacle is sensed and reacted to, so a fast sensing/position control loop is needed. Our experience in simulation suggests that a 1[KHz] rate of closed-loop control would be needed. However, if the low-level controller is an inverse dynamics controller, then the motor control feedback gains can be kept low in the swing phase, and in case of stumbling the system will be soft when hitting the obstacle.

Bibliography

Blind dog jake. https://www.youtube.com/watch?v=2-wIvypLZ4Q. Accessed: 2014-11-12.

Controller mate software. http://www.orderedbytes.com/controllermate/, 2014. Accessed: 2014-11-27.

Merriam-webster online dictionary. http://www.merriam-webster.com/dictionary/multiped, 2014a. Accessed: 2014-11-11.

Merriam-webster online dictionary. http://www.merriam-webster.com/dictionary/reflex, 2014b. Accessed: 2014-11-24.

Mostafa Ajallooeian, Majid Nili Ahmadabadi, Babak Nadjar Araabi, and Hadi Moradi. Design, implementation and analysis of an alternation-based central pattern generator for multidimensional trajectory generation. *Robotics and Autonomous Systems*, 60(2):182–198, 2012.

R McN Alexander. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 3(2):49–59, 1984.

Paolo Arena, Luigi Fortuna, Mattia Frasca, and Giovanni Sicurella. An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(4):1823–1837, 2004.

Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson R De Pieri, and Darwin G Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2554–2561. IEEE, 2013.

Stéphane Bazeille, Victor Barasuol, Michele Focchi, Ioannis Havoutis, Marco Frigerio, Jonas Buchli, Claudio Semini, and Darwin G Caldwell. Vision enhanced reactive locomotion control for trotting on rough terrain. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

Karsten Berns, Winfried Ilg, M Deck, J Albiez, and Rüdiger Dillmann. Mechanical construction and computer architecture of the four-legged walking machine bisam. *IEEE/ASME transactions on mechatronics*, 4(1):32–38, 1999.

Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, page 17, 2013.

Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008.

Lise Broch, Rey D Morales, Anthony V Sandoval, and Michael S Hedrick. Regulation of the respiratory central pattern generator by chloride-dependent inhibition during development in the bullfrog (rana catesbeiana). *Journal of Experimental Biology*, 205(8):1161–1169, 2002.

Jonas Buchli and Auke Jan Ijspeert. Self-organized adaptive legged locomotion in a compliant quadruped robot. *Autonomous Robots*, 25(4):331–347, 2008.

Jonas Buchli, Fumiya Iida, and Auke Jan Ijspeert. Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3903–3909. IEEE, 2006a.

Jonas Buchli, Ludovic Righetti, and Auke Jan Ijspeert. Engineering entrainment and adaptation in limit cycle systems. *Biological Cybernetics*, 95(6):645–664, 2006b.

Jonas Buchli, Mrinal Kalakrishnan, Michael Mistry, Peter Pastor, and Stefan Schaal. Compliant quadruped locomotion over rough terrain. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 814–820. IEEE, 2009.

Avis H Cohen, Philip J Holmes, and Richard H Rand. The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: a mathematical model. *Journal of mathematical biology*, 13(3):345–369, 1982.

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel Van De Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics (TOG)*, 30(4):59, 2011.

Alessandro Crespi and Auke Jan Ijspeert. Amphibot ii: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th international conference on climbing and walking robots (CLAWAR 2006)*, number BIOROB-CONF-2006-001, pages 19–27, 2006.

Monica A Daley and Andrew A Biewener. Running over rough terrain reveals limb control for intrinsic stability. *Proceedings of the National Academy of Sciences*, 103(42):15681–15686, 2006.

Monica A Daley, James R Usherwood, Gladys Felix, and Andrew A Biewener. Running over rough terrain: guinea fowl maintain dynamic stability despite a large unexpected change in substrate height. *Journal of Experimental Biology*, 209(1):171–187, 2006.

Sarah Degallier and Auke Ijspeert. Modeling discrete and rhythmic movements through motor primitives: a review. *Biological cybernetics*, 103(4):319–338, 2010.

Sarah Dégallier, Cristina P Santos, Ludovic Righetti, and Auke Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 512–517. IEEE, 2006.

Thomas E Dick, Yoshitaka Oku, J Richard Romaniuk, and Neil S Cherniack. Interaction between central pattern generators for breathing and swallowing in the cat. *The Journal of physiology*, 465(1):715–730, 1993.

Kenji Doya and Shuji Yoshizawa. Adaptive neural oscillator using continuous-time back-propagation learning. *Neural Networks*, 2(5):375–385, 1989.

Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot. *The International Journal of Robotics Research*, 27(2):213–228, 2008.

Johannes Ernesti, Ludovic Righetti, Martin Do, Tamim Asfour, and Stefan Schaal. Encoding of periodic and their transient motions by a single dynamic movement primitive. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 57–64. IEEE, 2012.

Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

Tamar Flash and Binyamin Hochner. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology*, 15(6):660–666, 2005.

Michele Focchi, Victor Barasuol, Ioannis Havoutis, Jonas Buchli, Claudio Semini, and Darwin G Caldwell. Local reflex generation for obstacle negotiation in quadrupedal locomotion. In *Int. Conf. on climbing and walking robots (CLAWAR)*, 2013.

H Forssberg. Stumbling corrective reaction: a phase-dependent compensatory reaction during locomotion. *J Neurophysiol*, 42(4):936–953, 1979.

Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

Yasuhiro Fukuoka, Hiroshi Kimura, and Avis H Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202, 2003.

William Fulton. Algebraic topology: a first course. 1995.

Andrej Gams, Sarah Degallier, AJ Ijspeert, and Jadran Lenarcic. Dynamical system for learning the waveform and frequency of periodic signals–application to drumming. In *Proceedings of the 17th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD2008)*. Citeseer, 2008.

Sébastien Gay, José Santos-Victor, and Auke Ijspeert. Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 194–201. Ieee, 2013.

Hartmut Geyer and Hugh Herr. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(3):263–273, 2010.

RC Griggs, RF Jozefowicz, and MJ Aminoff. Approach to the patient with neurologic disease. *Cecil Medicine. 24th ed. Philadelphia, Pa: Saunders Elsevier*, 2011.

Sten Grillner, Örjan Ekeberg, Abdeljabbar El Manira, Anders Lansner, David Parker, Jesper Tegner, and Peter Wallen. Intrinsic function of a neuronal network—a vertebrate central pattern generator. *Brain Research Reviews*, 26(2):184–197, 1998.

John Guckenheimer and Philip Holmes. Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. 1983.

JM Halbertsma. *The stride cycle of the cat.* Blackwell Scientific Publ., 1983.

Nalin Harischandra, Jean-Marie Cabelguen, and Örjan Ekeberg. A 3d musculo-mechanical model of the salamander for the study of different gaits and modes of locomotion. *Frontiers in neurorobotics*, 4, 2010.

EM Hasler, W Herzog, TR Leonard, A Stano, and H Nguyen. In vivo knee joint loading and kinematics before and after acl transection in an animal model. *Journal of biomechanics*, 31 (3):253–262, 1997.

Ioannis Havoutis, Jesus Ortiz, Stephane Bazeille, Victor Barasuol, Claudio Semini, and Darwin G Caldwell. Onboard perception-based trotting and crawling with the hydraulic quadruped robot (hyq). In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 6052–6057. IEEE, 2013.

Jessica K Hodgins and MARCH Raibert. Adjusting step length for rough terrain locomotion. *Robotics and Automation, IEEE Transactions on*, 7(3):289–298, 1991.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *Siam Review*, 48(2):207–304, 2006.

Scott L Hooper. Central pattern generators. *eLS*, 2001.

Eberhard Hopf. Abzweigung einer periodischen lösung von einer stationären lösung eines differentialsystems. *Ber. Math.-Phys. Kl Sächs. Akad. Wiss. Leipzig*, 94:1–22, 1942.

Marco Hutter, Christian Gehring, MA Hopflinger, M Blosch, and Roland Siegwart. Toward combining speed, efficiency, versatility, and robustness in an autonomous quadruped.

Marco Hutter, Christian Gehring, Michael Bloesch, MA Hoepflinger, C David Remy, and R Siegwart. Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *15th International Conference on Climbing and Walking Robot-CLAWAR 2012*, number EPFL-CONF-181042, 2012.

Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

Auke Jan Ijspeert and Alessandro Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In *Proceedings of the 2007 IEEE international conference on robotics and automation (ICRA 2007)*, number BIOROB-CONF-2007-009, pages 262–268, 2007.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IROS*, pages 958–963, 2002a.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1398–1403. IEEE, 2002b.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, pages 1547–1554, 2003.

Auke Jan Ijspeert, Alessandro Crespi, Dimitri Ryczko, and Jean-Marie Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315 (5817):1416–1420, 2007.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

Winfried Ilg, Jan Albiez, H Jedele, Karsten Berns, and Rüdiger Dillmann. Adaptive periodic movement control for the four legged walking machine bisam. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 2354–2359. IEEE, 1999.

Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. Cpg model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(3):171–179, 2003.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002.

Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

Guillaume Jouffroy. Design of oscillatory recurrent neural network controllers with gradient based algorithms. In *ESANN*, pages 7–12, 2008.

Chia-Feng Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):997–1006, 2004.

Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, and Stefan Schaal. Learning locomotion over rough terrain using terrain templates. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 167–172. IEEE, 2009.

Konstantinos Karakasiliotis. Legged locomotion with spinal undulations, 2013.

Konstantinos Karakasiliotis, Nadja Schilling, Jean-Marie Cabelguen, and Auke Jan Ijspeert. Where are we in understanding salamander locomotion: biological and robotic perspectives on kinematics. *Biological cybernetics*, 107(5):529–544, 2013.

Hassan K Khalil and JW Grizzle. Nonlinear systems. 2002.

Ole Kiehn and Simon JB Butt. Physiological, anatomical and genetic identification of cpg neurons in the developing mammalian spinal cord. *Progress in neurobiology*, 70(4):347–361, 2003.

Hiroshi Kimura, Yasuhiro Fukuoka, and Avis H Cohen. Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *The International Journal of Robotics Research*, 26(5):475–490, 2007a.

Hiroshi Kimura, Yasuhiro Fukuoka, and Avis H Cohen. Biologically inspired adaptive walking of a quadruped robot. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):153–170, 2007b.

J Zico Kolter and Andrew Y Ng. The stanford littledog: A learning and rapid replanning approach to quadruped locomotion. *The International Journal of Robotics Research*, 30(2): 150–174, 2011.

Kana Kotaka, Barkan Ugurlu, Michihiro Kawanishi, and Tatsuo Narikiyo. Prototype development and real-time trot-running implementation of a quadruped robot: Robocat-1. In *Mechatronics (ICM), 2013 IEEE International Conference on*, pages 604–609. IEEE, 2013.

Ivana Kovacic and Michael J Brennan. The duffing equation: Nonlinear oscillators and their behaviour. 2011.

Arthur D Kuo. The relative roles of feedforward and feedback in the control of rhythmic movements. *MOTOR CONTROL-CHAMPAIGN-*, 6(2):129–145, 2002.

Yasuaki Kuroe and H Lima. A learning method for synthesizing spiking neural oscillators. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3882–3886. IEEE, 2006.

Yasuaki Kuroe and Kei Miura. A method of oscillatory trajectory generation using recurrent hybrid neural networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 706–711. IEEE, 2005.

Edouard Leclercq, Fabrice Druaux, Dimitri Lefebvre, and Salem Zerkaoui. Autonomous learning algorithm for fully connected recurrent networks. *Neurocomputing*, 63:25–44, 2005.

Guang Lei Liu, Maki K Habib, Keigo Watanabe, and Kiyotaka Izumi. Central pattern generators based on matsuoka oscillators for the locomotion of biped robots. *Artificial Life and Robotics*, 12(1-2):264–269, 2008.

GE Loeb. Control implications of musculoskeletal mechanics. In *Engineering in Medicine and Biology Society, 1995., IEEE 17th Annual Conference*, volume 2, pages 1393–1394. IEEE, 1995.

Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

Mantas LukošEvičIus and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

Marilyn MacKay-Lyons. Central pattern generation of locomotion: a review of the evidence. *Physical therapy*, 82(1):69–83, 2002.

Vítor Matos and Cristina P Santos. Omnidirectional locomotion in a quadruped robot: A cpg-based approach. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3392–3397. IEEE, 2010.

Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological cybernetics*, 52(6):367–376, 1985.

Kiyotoshi Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological cybernetics*, 56(5-6):345–353, 1987.

Kiyotoshi Matsuoka. Analysis of a neural oscillator. *Biological cybernetics*, 104(4-5):297–304, 2011.

Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In *Virtual Worlds*, pages 254–263. Springer, 1998.

Olivier Michel. Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*, 2004.

Ross H Miller, Scott CE Brandon, and Kevin J Deluzio. Predicting sagittal plane biomechanics that minimize the axial knee joint contact force during walking. *Journal of biomechanical engineering*, 135(1):011007, 2013.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

Albert Mukovskiy, Jean-Jacques E Slotine, and Martin A Giese. Dynamically stable control of articulated crowds. *Journal of Computational Science*, 4(4):304–310, 2013.

Michael P Murphy, Aaron Saunders, Cassie Moreira, Alfred A Rizzi, and Marc Raibert. The littledog robot. *The International Journal of Robotics Research*, page 0278364910387457, 2010.

Ferdinando A Mussa-Ivaldi, Simon F Giszter, and Emilio Bizzi. Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of Sciences*, 91(16): 7534–7538, 1994.

Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2):79–91, 2004.

Bojan Nemec and Ales Ude. Action sequencing using dynamic movement primitives. *Robotica*, 30(5):837–846, 2012.

Peter D Neuhaus, Jerry E Pratt, and Matthew J Johnson. Comprehensive summary of the institute for human and machine cognition's experience with littledog. *The International Journal of Robotics Research*, 30(2):216–235, 2011.

Arne Nordmann, Alexandre Tuleu, and Sebastian Wrede. A domain-specific language and simulation architecture for the oncilla robot. In *ICRA 2013 Workshop on Developments of Simulation Tools for Robotics & Biomechanics*, 2013.

Masafumi Okada, Koji Tatani, and Yoshihiko Nakamura. Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1410–1415. IEEE, 2002.

Dai Owaki, Takeshi Kano, Ko Nagasawa, Atsushi Tero, and Akio Ishiguro. Simple robot suggests physical interlimb communication is essential for quadruped walking. *Journal of The Royal Society Interface*, 10(78):20120669, 2013.

Barak A Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989.

Barak A Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *Neural Networks, IEEE Transactions on*, 6(5):1212–1228, 1995.

Claude Perret and Jean-Marie Cabelguen. Main characteristics of the hindlimb locomotor cycle in the decorticate cat with special reference to bifunctional muscles. *Brain research*, 187(2):333–352, 1980.

Krešimir Petrinec and Z Kovacic. Trajectory planning algorithm based on the continuity of jerk. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–5. IEEE, 2007.

Arkady Pikovsky, Michael Rosenblum, and Jürgen Kurths. Synchronization: a universal concept in nonlinear sciences. 2003.

Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

James Pippine, Douglas Hackett, and Adam Watson. An overview of the defense advanced research projects agency's learning locomotion program. *The International Journal of Robotics Research*, 30(2):141–144, 2011.

R Playter, M Buehler, and M Raibert. Bigdog. In *Defense and Security Symposium*, pages 62302O–62302O. International Society for Optics and Photonics, 2006.

Ion R Popescu and William N Frost. Highly dissimilar behaviors mediated by a multifunctional network in the marine mollusk tritonia diomedea. *The Journal of neuroscience*, 22(5): 1985–1993, 2002.

Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207. IEEE, 2006.

Dale Purves. *Neuroscience: Third Edition*. Massachusetts, Sinauer Associates, Inc., 2004.

Marc Raibert. Dynamic legged robots for rough terrain. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 1–1. IEEE, 2010.

Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, et al. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress*, pages 10823–10825, 2008.

Marc H Raibert et al. *Legged robots that balance*, volume 3. MIT press Cambridge, MA, 1986.

Ludovic Righetti and Auke Jan Ijspeert. Design methodologies for central pattern generators: An application to crawling humanoids. In *Robotics: Science and Systems*, 2006a.

Ludovic Righetti and Auke Jan Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1585–1590. IEEE, 2006b.

Ludovic Righetti and Auke Jan Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 819–824. IEEE, 2008.

Ludovic Righetti, Jonas Buchli, and Auke Jan Ijspeert. Dynamic hebbian learning in adaptive frequency oscillators. *Physica D: Nonlinear Phenomena*, 216(2):269–281, 2006.

Renaud Ronsse, Nicola Vitiello, Tommaso Lenzi, Jesse van den Kieboom, Maria Chiara Carrozza, and Auke Jan Ijspeert. Adaptive oscillators with human-in-the-loop: Proof of concept for assistance and rehabilitation. In *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pages 668–674. IEEE, 2010.

Renaud Ronsse, Nicola Vitiello, Tommaso Lenzi, Jesse van den Kieboom, Maria Chiara Carrozza, and Auke Jan Ijspeert. Human–robot synchrony: flexible assistance using adaptive oscillators. *Biomedical Engineering, IEEE Transactions on*, 58(4):1001–1012, 2011.

Serge Rossignol, Réjean Dubuc, and Jean-Pierre Gossard. Dynamic sensorimotor interactions in locomotion. *Physiological reviews*, 86(1):89–154, 2006.

A Ruiz, David H Owens, and Stuart Townley. Existence, learning, and replication of periodic motions in recurrent neural networks. *IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council*, 9(4):651–661, 1997.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

Stefan Schaal and Christopher G Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks. p. 471-482 2007*, pages 471–482, 2007.

Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer, 2000.

Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, page 0959651811402275, 2011.

Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey Lang, and Sangbae Kim. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3307–3312. IEEE, 2013.

ML Shik, FV Severin, and ORLOVSKI. GN. Control of walking and running by means of electrical stimulation of mid-brain. *BIOPHYSICS-USSR*, 11(4):756, 1966.

Alexander Shkolnik, Michael Levashov, Ian R Manchester, and Russ Tedrake. Bounding on rough terrain with the littledog robot. *The International Journal of Robotics Research*, page 0278364910388315, 2010.

Russell Smith. Ode: Open dynamics engine. *Online at: http://www.ode.org*, 2003.

Andrea Soltoggio and Jochen J Steil. How rich motor skills empower robots at last: Insights and progress of the amarsi project. *KI-Künstliche Intelligenz*, 26(4):407–410, 2012.

Alexander Sproewitz, Rico Moeckel, Jérôme Maye, and Auke Jan Ijspeert. Learning to move in modular robots using central pattern generators and online optimization. *The International Journal of Robotics Research*, 27(3-4):423–443, 2008.

Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950, 2013.

Steven H Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1):1–20, 2000.

Gentaro Taga, Yoko Yamaguchi, and Hiroshi Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological cybernetics*, 65 (3):147–159, 1991.

Gerald Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.

Kurt A Thoroughman and Reza Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407(6805):742–747, 2000.

Stuart Townley, Achim Ilchmann, Martin G Weiß, Warren McClements, Antonio C Ruiz, David H Owens, and D Pratzel-Wolters. Existence and learning of oscillations in recurrent neural networks. *Neural Networks, IEEE Transactions on*, 11(1):205–214, 2000.

Alexandre Tuleu, Mostafa Ajallooeian, Alexander Spröwitz, Peter Löpelmann, and A Ijspeert. Trot gait locomotion of a cat sized quadruped robot. In *International Workshop on BioInspired Robots*, pages 1–4, 2011.

Barkan Ugurlu, Kana Kotaka, and Tatsuo Narikiyo. Actively-compliant locomotion control on rough terrain: Cyclic jumping and trotting experiments on a stiff-by-nature quadruped. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3313–3320. IEEE, 2013.

Daniel E Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 1969.

David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howardy, Alfred A Rizzi, and Marc Raibert. Autonomous navigation for bigdog. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4736–4741. IEEE, 2010.

Xiaodong Wu and Shugen Ma. Cpg-based control of serpentine locomotion of a snake-like robot. *Mechatronics*, 20(2):326–334, 2010.

Francis wyffels and Benjamin Schrauwen. Design of a central pattern generator using reservoir computing for learning human motion. In *Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL'09.*, pages 118–122. IEEE, 2009.

Francis wyffels, Jiwen Li, Tim Waegeman, Benjamin Schrauwen, and Herbert Jaeger. Frequency modulation of large oscillatory neural networks. *Biological Cybernetics*, 108(2):145–157, 2014. ISSN 0340-1200. doi: 10.1007/s00422-013-0584-0. URL `http://dx.doi.org/10.1007/s00422-013-0584-0`.

Wei Zhao, Junzhi Yu, Yimin Fang, and Long Wang. Development of multi-mode biomimetic robotic fish based on central pattern generator. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3891–3896. IEEE, 2006.

Curriculum Vitae

Personal Data

Name	Mostafa Ajallooeian
Birth	Iran, December 21th, 1983
Nationality	Iranian
Address	EPFL STI IBI BIOROB
	Station 14, INN 235
	CH-1015 Lausanne, Switzerland
Phone	+41 21 693 26 76
Fax	+41 21 693 37 05
Email	mostafa.ajallooeian@epfl.ch

Education

2010-2015	PhD Candidate, Biorobotics Laboratory, EFPL, Switzerland
2006-2009	MSc, Machine Intelligence and Robotics, University of Tehran, Iran
2002-2006	BSc, Computer Engineering - Software, Kashan University, Iran

Publications

Journal Articles

[5] A. Sprowitz, **M. Ajallooeian**, A. Tuleu and A. Ijspeert. Kinematic Primitives for Walking and Trotting Gaits of a Quadruped Robot with Compliant Legs, in Frontiers in Computational Neuroscience, vol. 8, num. 27, p. 1-13, 2014.

[4] **M. Ajallooeian**, J. van den Kieboom, A. Mukovskiy, M. Giese and A. Ijspeert. A General Family of Morphed Nonlinear Phase Oscillators with Arbitrary Limit Cycle Shape, in Physica D: Nonlinear Phenomena, vol. 263, p. 41-56, 2013.

[3] A. Sproewitz, A. Tuleu, M. Vespignani, **M. Ajallooeian** and E. Badri et al. Towards Dynamic Trot Gait Locomotion—Design, Control and Experiments with Cheetah-cub, a Compliant Quadruped Robot, in International Journal of Robotics Research, vol. 32, num. 8, p. 932 - 950, 2013.

[2] **M. Ajallooeian**, M. N. Ahmadabadi, B. N. Araabi and H. Moradi. Design, Implementation and Analysis of an Alternation-based Central Pattern Generator for Multidimensional Trajectory Generation, in Robotics and Autonomous Systems, vol. 60, num. 2, p. 182-198, 2012.

[1] G. G. Amiri, A. Shahjouei, S. Saadat and **M. Ajallooeian**. Hybrid Evolutionary-Neural Network Approach in Generation of Artificial Accelerograms Using Principal Component Analysis and Wavelet-Packet Transform, in Journal Of Earthquake Engineering, vol. 15, p. 50-76, 2011.

Conference Appearances

[13] **M. Ajallooeian**, A. Tuleu, A. Sprowitz, P. Eckert and M. Vespignani et al. Rich Locomotion Skills with the Oncilla Robot. Dynamic Walking 2014, ETHZ, Zurich, Switzerland, 2014.

[12] **M. Ajallooeian**, S. Gay, A. Tuleu, A. Sprowitz and A. Ijspeert. Modular Control of Limit Cycle Locomotion over Unperceived Rough Terrain. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 2013.

[11] **M. Ajallooeian**, S. Pouya, A. Sproewitz and A. Ijspeert. Central Pattern Generators Augmented with Virtual Model Control for Quadruped Rough Terrain Locomotion. 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), Karlsruhe, Germany, 2013.

[10] **M. Ajallooeian**, S. Pouya, S. Gay, A. Tuleu and A. Sprowitz et al. Towards Modular Control for Moderately Fast Locomotion over Unperceived Rough Terrain. Dynamic Walking 2013, Carnegie Mellon University, Pittsburgh, USA, 2013.

[9] **M. Ajallooeian**, A. Sproewitz, A. Tuleu and A. Ijspeert. Data-driven Extraction of Drive Functions for Legged Locomotion: A Study on Cheetah-cub Robot. 6th International Conference on Adaptive Motion of Animals and Machines, 2013, Darmstadt, Germany, 2013.

[8] S. Pouya, **M. Ajallooeian** and A. Ijspeert. A Closed-Loop Optimal Control Approach for Online Control of A Planar Monopod Hopper. 15th International Conference on Climbing and walking Robots (CLAWAR), Maryland, USA, 2012.

[7] A. Sprowitz, A. Tuleu, M. Vespignani, **M. Ajallooeian** and E. Badri. Robot Trotting with Segmented Legs in Simulation and Hardware.. Dynamic Walking 2012, Pensacola Beach, Florida. USA, 2012.

[6] A. Sprowitz, L. Kuechler, A. Tuleu, **M. Ajallooeian** and M. D'Haene et al. Oncilla robot: A Light-weight Bio-inspired Quadruped Robot for Fast Locomotion in Rough Terrain. 5th International Symposium on Adaptive Motion of Animals and Machines, Awaji, Japan, 2011.

[5] A. Tuleu, **M. Ajallooeian**, A. Spröwitz, P. Loepelmann and A. Ijspeert. Trot Gait Locomotion of A Cat Sized Quadruped Robot. International Workshop on Bio-inspired Robots, Nantes,

France, 2011.

[4] H. Hajimirsadeghi, M. Nili Ahmadabadi, **M. Ajallooeian**, B. Nadjar Araabi and H. Moradi. Conceptual Imitation Learning: Application to Human-Robot Interaction. 2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, 2010.

[3] **M. Ajallooeian**, A. Borji, B. N. Araabi, M. N. Ahmadabadi and H. Moradi. Fast Hand Gesture Recognition based on Saliency Maps: An Application to Interactive Robotic Marionette Playing. 18th IEEE International Symposium on Robot and Human Interactive Communication, Toyama, JAPAN, 2009.

[2] **M. Ajallooeian**, M. N. Ahmadabadi, B. N. Araabi and H. Moradi. An Imitation Model based on Central Pattern Generator with Application in Robotic Marionette Behavior Learning. IEEE RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009.

[1] S. A. Amraii, **M. Ajallooeian** and C. Lucas. A Dynamic Fuzzy-based Crossover Method for Genetic Algorithms. 19th IEEE International Conference on Tools with Artificial Intelligence, Patras, GREECE, Proceedings - International Conference On Tools With Artificial Intelligence, 2007.